Módszer köztes tárolókat nem tartalmazó szakaszos működésű rendszerek ütemezésére

Doktori (PhD) értekezés

Holczinger Tibor

Témavezető: Dr. Friedler Ferenc

Veszprémi Egyetem

Informatikai Tudományok

Doktori Iskola

VESZPRÉM

2004

MÓDSZER KÖZTES TÁROLÓKAT NEM TARTALMAZÓ SZAKASZOS MŰKÖDÉSŰ RENDSZEREK ÜTEMEZÉSÉRE

Értekezés doktori (PhD) fokozat elnyerése érdekében

Írta:
Holczinger Tibor

Készült a Veszprémi Egyetem Informatikai Tudományok Doktori Iskolája keretében

Témavezető: Dr. Friedler Ferenc

Elfogadásra javasolom (igen / nem)

.....
Dr. Friedler Ferenc

A jelölt a doktori szigorlaton ..... %-ot ért el.

Veszprém,

.....
a szigorlati bizottság elnöke

Az értekezést bírálóként elfogadásra javasolom:

Első bíráló (Dr.) igen / nem

.....
aláírás

Második bíráló (Dr.) igen / nem

.....
aláírás

A jelölt az értekezés nyilvános vitáján ..... %-ot ért el.

Veszprém,

.....
a Bíráló Bizottság elnöke

A doktori (PhD) oklevél minősítése .....

.....
Az EDT elnöke

Tartalomjegyzék

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Friedler Ferencnek irányításáért és értékes szakmai tanácsaiért. Köszönettel tartozom a Department of Chemical Engineering, Universitat Politecnica de Catalunya munkatársainak különösen Luis Puigjanernek professzornak és Javier Romero PhD hallgatónak a szakmai együttműködésükért. Köszönet illeti továbbá a Veszprémi Egyetem Számítástudomány Alkalmazása Tanszék munkatársait, akik munkám elvégzéséhez bíztatást és támogatást nyújtottak. Külön szeretnék még köszönetet mondani családomnak kitartó támogatásukért.

Kivonat

Módszer köztes tárolókat nem tartalmazó szakaszos működésű rendszerek ütemezésére

Ütemezési feladatok széles körben fordulnak elő az informatikai rendszerekben, a termelő iparban (pl. a vegyiparban, az olajiparban, a gépiparban) a mezőgazdaságban és az építőiparban. Ezért gyakorlati szempontból fontos az optimális vagy közel optimális megoldások meghatározására alkalmas módszerek kifejlesztése.

A dolgozat olyan gráf leíráson (S-gráf) alapul, amely alkalmas általános ütemezési feladatok speciális tulajdonságait megfelelően kifejező leírására. A szerző egy, az S-gráfhoz illesztett szétválasztás és korlátozás algoritmust használ, amely képes a feladatok hatékony megoldására.

A dolgozatban a szerző vizsgálja az S-gráf használata során felmerülő kérdéseket. Bemutatja a feladat korrekt felírásához szükséges speciális S-gráf, az úgynevezett recept-gráf felépítésének menetét. Matematikai módszerekkel megadja, hogy egy S-gráf hogyan és mikor reprezentál egy megoldást valamint definiálja az ehhez szükséges fogalmakat.

A szerző kidolgozott az S-gráf leírás alapalgoritmusához egy kiterjesztést, amelynek segítségével jól lehet kezelni azokat a feladatokat, amelyekben a termékek előállításához több batch-re van szükség. A szerző továbbá készített az algoritmushoz két általánosan használható gyorsítási módszert, amelyek hatékonyságát egy szakirodalmi példán keresztül mutatja be, összehasonlítva a példához kifejlesztett módszerrel. Majd egy, a szakirodalomban igen nagynak számító ipari feladat megoldásán keresztül szemlélteti a módszer fontosságát.

Method for scheduling non-intermediate storage batch process systems

Scheduling problems appear in informatics systems, in production industry (chemical industry, oil industry, mechanical industry), in agriculture and in building industry. Hence, it is important to develop methods to determine the optimal or near optimal solutions.

The thesis is based on a graph (S-graph) representation which can illustrate appropriately the specific properties of scheduling problems. The author uses an S-graph based branch and bound algorithm that can solve the model efficiently.

In the thesis, the author examines the questions of using S-graphs. He introduces the steps of building of a special S-graph (so-called recipe-graph), which is necessary for the exact inscription of the problem. He determines when an S-graph represents a solution and he gives the necessary definitions.

The author extends the S-graph representation to the case when a product has more than one batches. He makes two general acceleration procedures for the algorithm and he illustrates of their efficiency by a literature example compared with the method developed for the example. Finally, he demonstrates the importance of the method by an industrial example, which is reckoned as big in the literature.

Methode zu Taktaufgaben von Systemen mit intermittierender Funktion, die keinen Zwischensammer enthalten

Es kommen in den Informatiksystemen Taktaufgaben ausgebreitet vor, in der Produktionsindustrie, (zum beispiele: in der Chemieindustrie, Ölindustrie, Maschinenindustrie) in der Landwirtschaft und in der Bauindustrie. Deshalb ist vom praktischen Standpunkt aus wichtig, die Methoden zu entwickeln, die zu der Bestimmung der optimalen oder fast optimalen Lösungen brauchen.

Die Arbeit beruht sich auf eine Beschreibung von Graf, die geeignet zu einer Beschreibung von Taktaufgaben ist, die speziellen Eigenschaften ausdrücken. Der Verfasser benutzt einen Trennung- und Beschränkungsalgorithmus, den man zu dem S-Graf fügen kann, und dieser Algorithmus kann die Aufgaben wirksamvoll lösen.

Der Verfasser untersucht in der Arbeit die Fragen, die mit der Benutzung von S-Graf auftauchen. Der Verfasser präsentiert den Gang, wie man den Sog. Rezept-Graf aufbauen müsste, der zu der korrekten Lösung der Aufgabe wichtig ist. Er gibt mathematischen Methoden dazu, wie und wann ein S-Graf eine Lösung repräsentiert, und definiert die dazu nötigen Definitionen.

Der Verfasser hat eine Ausdehnung ausgearbeitet, die zu dem Grundalgorithmus der Beschreitung der S-Graf nötig ist, damit kann man die Aufgaben gut lösen, wo man zu der Herstellung der Ware mehr Reiche braucht. Der Verfasser hat noch zwei Methoden zu dem Algorithmus ausgearbeitet, die alles Schneller machen, und deren Wirksamkeit er durch einen Beispiel veranschaulicht, und dann vergleicht er mit der zu dem Beispiel entwickelten Methode. Am Ende veranschaulichte der Verfasser die Wichtigkeit der Methode durch eine Lösung von einer bedeutenden industriellen Aufgabe.

1. Bevezetés

Ütemezési feladatok széles körben fordulnak elő termelő rendszerekben, például a vegyiparban, olajiparban, gépiparban, mezőgazdaságban és építőiparban. Mivel egy ütemezési feladat megoldásai a teljes rendszer működési idejére vonatkozóan nagymértékben eltérhetnek egymástól, ezért fontos a megoldások közül a legjobbat megtalálni, valamint az is elengedhetetlen, hogy az ütemezési feladatokat a felhasználó számára elfogadható időn belül meg lehessen oldani.

Egy ütemezési feladat többféleképen definiálható. A dolgozatban ütemezési feladatnak nevezzük azt a feladatot, ahol a lehető legrövidebb idő alatt kell adott mennyiségű termékeket előállítani a rendelkezésre álló szakaszos működésű berendezések felhasználásával, ahol egy terméket taszkok adott sorrendű végrehajtásával lehet előállítani. A gyakorlatban egy taszk elvégzésére több berendezés is alkalmas lehet, a közülük való választás is az ütemezési feladathoz tartozik.

Az eddig ismert megoldási módszerek egy része a gyakorlati tapasztalatoknak megfelelően megfogalmazott heurisztikus szabályokon alapul, melyek fő hátránya, hogy nem tudják biztosítani az optimumot. Egy másik napjainkban használatos módszer, hogy minden ütemezési feladatosztályhoz felírnak egy speciális matematikai modellt, és azt általános célú megoldó szoftverekkel oldják meg, amely nagyfokú kombinatorikus bonyolultság miatt erősen korlátozza a feladatok méretét. Az előbbiekből látszik, hogy minden feladatosztályhoz egy specializált algoritmusra és megoldó szoftverre van szükség, amelyhez jó kiindulási pontot nyújt az S-gráfhoz (Sanmartí és társai, 1998) illesztett korlátozás és szétválasztás módszeren alapuló algoritmus (Sanmartí és társai, 2002). Ehhez az alapalgoritmushoz készíthetőek feladatspecifikus valamint általánosan használható eljárások illetve gyorsítások, amelyeket könnyen illeszthetünk a meglévő algoritmushoz, és amelyek adott feladatosztályok speciális tulajdonságait figyelembe veszik és kihasználják. A feladat előzetes vizsgálata során eldönthető, hogy ezek közül az eljárások közül melyeket kell, illetve melyeket érdemes használni a gyorsabb megoldás

érdekében. Ezenkívül az algoritmus rugalmassága lehetővé teszi, hogy a felhasználó által kért követelményeket figyelembe lehessen venni, például az első három legjobb megoldás megadása vagy speciális heurisztikák beépítése.

A szakirodalomban eddig közölt vizsgálatokban nem fordítottak kellő hangsúlyt arra, hogy az ütemezési feladatoknak létezik-e általános, jól használható leírása, amely segítségével az ütemezés szinte minden feladatosztályát meg lehet oldani. Szakaszos működésű rendszerek ütemezéséhez egy jól használható, hatékony gráf leírást (S-gráf) és a hozzá elkészített algoritmust használtunk. Tekintettel arra, hogy ezt a módszertant későbbi felhasználásra szánjuk, a további kiterjesztések és gyorsítások hatékony fejlesztése érdekében szigorú formalizmust vezettünk be.

2. Szakirodalmi áttekintés

Ütemezési feladatoknál általában a cél a taszkok valamilyen szempontból optimális működési sorrendjének meghatározása a meglévő erőforrások felhasználásával. Az ipari méretű ütemezési feladatok esetén rendszerint nagy számú, különböző típusú és minőségű terméket állítanak elő, továbbá a költségek csökkentése érdekében figyelembe vehetők a termékek különböző előállítási lehetőségei is. Ez a nagy fokú rugalmasság az egyik fő forrása az ütemezési feladatok bonyolultságának. A gyakorlatban egy ütemezési feladat matematikai modelljének megfelelő leírásához rengeteg változót kell bevezetni (például a berendezések taszkokhoz való rendeléséhez, a termelés és a taszkok sorrendjének leírásához, a taszkok időzítéséhez), amely modell nehezen vagy gyakran egyáltalán nem oldható meg általános célú megoldó szoftverekkel.

Az általános ütemezési feladat első szisztematikus modelljét Sparrow és társai (1975) adták meg egy vegyes egész nemlineáris programozási (*mixed integer nonlinear programming*, MINLP) feladattal. Ebben a modellben a termékek teljes gyártási idejét úgy modellezték, hogy minden egyes termékre figyelembe vették az előállítandó termékmennyiséghez szükséges batch-ek számát, azaz hogy hányszor kell a termék előállítási folyamatát végrehajtani. A terület kutatási témáinak egyik fő irányvonala volt a formalizmushoz optimumot biztosító, az ipari méretű feladatokat reális időn belül megoldó módszer megtalálása. Grossmann és Sargent (1979) erre a modellre írt fel egy geometriai feladatot és a Kuhn-Tucker feltételek segítségével bebizonyították az optimum globalitását. Azt is bemutatták, hogy meg lehet oldani ennek a feladatnak egy relaxált részfeladatát, amelyben nem veszik figyelembe a berendezések méretének diszkrétségét. Knopf és társai (1982) egy általánosabb esetet vizsgáltak fél-folytonos (*semicontinuous*) egységekkel, ahol modell szintén egy geometriai feladat volt, de ők a konvex primál alak (*convex primal form*) segítségével oldották meg. Ravemark és Rippin (1998) az eredeti Sparrow és társai (1975) által megadott formalizmust használták többfajta terméket előállító rendszerekhez (*multiproduct plant*) és egy logaritmikus transzformációval biztosították a MINLP modell konvexitását. Meg kell jegyezni, hogy a fent említett módszerek csak

nagyon kis méretű feladatok megoldására használhatóak. A feltételek egyszerűsítése, mint például a keresési tér csökkentése (lásd például Reklaitis, 1981), ahol a taszkok helyett a termékeket vagy batch-eket kell ütemezni, jó eredményhez vezethetnek, de az optimalitást a legtöbb esetben nem lehet garantálni.

Számos heurisztikákon alapuló módszert fejlesztettek ki speciális típusú többcélú (***multipurpose***) és többfajta terméket előállító szakaszos működésű rendszerekhez. A heurisztikán alapuló módszerek fő hátránya, hogy az optimum megtalálása nem biztosítható. Suhami és Mah (1982) egy gráf leíráson alapuló eljárást készítettek kis méretű többcélú rendszerek megoldásához. Módszerükben a lehetséges struktúrák közül heurisztika segítségével választják ki a legkisebb költségűt. Eljárásuk legfeljebb csak 7 terméket és 10 műveleti egység típust képes kezelni továbbá nem tudják ezzel a modellel kezelni a fél-folytonos műveleteket és a köztes tárolásokat. Tan és Mah (1998) egy olyan heurisztikán alapuló eljárást javasolt, amely megengedi a tervezés közbeni emberi beavatkozást. Az optimalizálást két lépésben oldották meg, első lépésben heurisztikus előrejelző modell segítségével meghatározzák a lehetséges konstrukciót, majd a második lépésben különböző tervezési lehetőségek figyelembevételével optimalizálják. Ezzel a modellel nem tudják kezelni azokat a recepteket, ahol egy taszknak több bemenete is lehet (***branched product recipe***) valamint azokat, ahol a termékek sorrendje változik egy munkaidény alatt (például: A, B, A, B, ..., sorrend az A és a B termékekre). Lee és Lee (1996) kidolgoztak egy módszert, amely kezelni tudja a különböző típusú párhuzamos egységeket és az úgynevezett „***out of phase***” műveleteket.

A szakaszos működésű kémiai folyamatok rövid távú ütemezése (***short-term scheduling***) az elmúlt két évtizedben nagy figyelemnek örvendett. Több különböző megközelítést, matematikai formalizmust és megoldó algoritmust készítettek. Jó áttekintés található ezekről a módszerekről, Pinto és Grossmann (1995), Shah (1998), Penky és Reklaitis (1998) és Puigjaner (1999) munkáiban.

A szakaszos működésű kémiai folyamatok rövid távú ütemezése nagy hasonlóságot mutat a kötegelt ütemezési feladattal (***job-shop problem***), amely az

operációkutatásban egy széles körben kutatott problémaosztály. Az egyik hagyományos megközelítés a gráf leíráson alapuló szétválasztás és korlátozás (*branch and bound*, B&B) algoritmus (lásd például, Adams és társai, 1988; Carlier és Pinson, 1989). A szétválasztás és korlátozás algoritmust gyakran használják speciális heurisztikákkal, amelyek nagy mértékben gyorsítják az algoritmus konvergenciáját és optimális, vagy közel optimális megoldást adnak. Valójában a szakaszos kémiai folyamatok ütemezése nagyban különbözik a kötegelt ütemezési feladatoktól. Ezek a feladatok általában bonyolultabbak mivel több feltételt kell figyelembe venni. Például az instabil köztes termékeket azonnal fel kell dolgozni, illetve folyékony halmazállapotú anyagok tárolása a gépiparban nem fordul elő. Ezek a különbségek kizárják, hogy a kötegelt ütemezéshez használt gráf leírást közvetlenül lehessen használni szakaszos vegyipari folyamatok ütemezéséhez, ezért szakaszos vegyipari rendszerek ütemezéséhez általában más módszereket használnak. A legelterjedtebb módszerek elsősorban a matematikai programozási modellek (például Voudouris és Grossmann, 1994; Sanmartí és társai, 1996), feladat-specifikus heurisztikákkal vagy sztochasztikus (például genetikus algoritmusok) módszerekkel (például, Kudva és társai, 1994; Graells és társai, 1996; Hasebe és társai, 1996; Murakami és társai, 1997) kiegészítve.

Összetett feladatok megoldása során nagyon fontos a megfelelő leírás kiválasztása (például STN leírás: Kondili és társai, 1993; RTN leírás: Shilling és Pantelides, 1996; EON leírás: Graells és társai, 1998), ugyanis a megfelelő leírás biztosíthatja a feladat egyedi tulajdonságainak kihasználhatóságát növelve az eljárás hatékonyságát. Továbbá, ha a leírás már maga kifejezi az optimalizálás szempontjából kritikus pontokat, ez hozzájárulhat a feladat megértéséhez, amely megkönnyítheti új algoritmusok kifejlesztését vagy a már meglévők továbbfejlesztését. Ezen kívül a választott leírás általánossága meghatározza a kezelhető feladatok halmazát is.

A szakaszos működésű kémiai folyamatok rövid távú ütemezéséhez Kondili és társai (1993) kidolgoztak egy vegyes egész lineáris programozási (*mixed integer linear programming*, MILP) modellt használva az állapot-taszk hálózat (*state-task network*, STN) leírást. Az állapotok (*state*) jelölik a

nyersanyagokat, a köztes termékeket és a végtermékeket, valamint a taszkok (*task*) jelölik a műveleteket, amelyekben az egyik állapot egy másikba alakul át. Ebben a formalizmusban a rendszer működéséhez rendelkezésre álló időt (*time horizon*) felosztják egyforma méretű darabokra, ahol a taszkok kezdési és befejezési idejének meg kell egyeznie ezen diszkrét időpontokkal. Ebben a modellben a fő előny a nagyon általános folyamatok, receptek kezelésének lehetősége, beleértve például az anyagok körforgását, a tárolási típusokat és az erőforrásokkal kapcsolatos feltételeket. A megközelítésben a legnagyobb problémát a létrehozott MILP modell nagy mérete illetve a diszkretizálás pontjainak a meghatározása jelenti. Ez a munka alapjául szolgált sok más kutatásnak, ahol a fő cél a számítási teljesítmény növelése volt, felhasználva a leírás előnyeit. Sahinidis és társai (1991) szétbontották a modellt úgy, hogy a modell nagyobb mérete ellenére növelték a megoldás hatékonyságát. Shah és társai (1993) a célfüggvény lehető legjobb közelítése érdekében változtattak a hozzárendelési szabályokon. Elkamel (1993) egy heurisztika segítségével szétbontotta a feladatot. Yee és Shah (1998) vizsgálta további heurisztikák beépítésének hatását.

Pantelides (1994) az STN leírás és a hozzá tartozó formalizmus helyett egy alternatív leírást ajánlott, az erőforrás-taszk hálózatot (*resource-task network*, RTN), amely az erőforrások egységes leírásán alapult. Ebben a leírásban a taszkok erőforrásokat (*resource*) fogyasztanak és állítanak elő nem anyagokat. Erőforrásnak minősülnek a nyersanyagok, a köztes termékek, a végtermékek, az energia, az emberi erőforrás, a tárolás és a szállítási eszközök. A taszkok definíciója megegyezik az STN leírásban használttal, kiegészítve a szállítással, a tisztítással és a tárolással.

Zhang és Sargent (1994) és Zhang (1995) az RTN leíráson alapuló folytonos idejű modell készített. Ebben a formalizmusban a teljes időtartam változó, előre meg nem határozott méretű részekre van feldarabolva, ahol a taszkok kezdő és végidőpontjai adják meg az időintervallumok szélességét. A matematikai formalizmus egy MINLP modellt eredményez, amelyet a Glover transzformációval (1975) linearizáltak és így egy MILP feladatot kaptak. Természetesen, mint minden linearizálási technika esetében, a feladat dimenziója

jelentősen megnőtt és így a feladat hamar elér ahhoz a mérethatárhoz, amely felett már nehezen kezelhető az általánosan használt MILP megoldókkal. Hasonló módszert dolgozott ki Schilling és Pantelides (1996) azzal a különbséggel, hogy ők feltételezték, hogy egy taszk mérete (a végrehajtási ideje vagy az adott idő alatt előállított anyag mennyisége) nem függ a választott berendezéstől, valamint kikötötték, hogy a taszkok csak a működésük elején és a végén vannak közvetlen kapcsolatban az erőforrásokkal. A kapott MINLP feladatot szintén a Glover transzformációval alakították át MILP feladattá, amit egy szétválasztás és korlátozás algoritmussal oldottak meg. Az algoritmus újdonsága abban rejlett, hogy nem csak az egész, hanem a folytonos változók szerint is végeztek szétválasztást.

Mockus és Reklaitis (1997) az STN leíráson alapuló folytonos idejű modellt készített bemutatva az időintervallumok és az események koncepcióját. Modelljükben az események reprezentálják a taszkok kezdetét és a végét, valamint az időintervallumok a két esemény között eltelt idő hosszát. A feladatot egy vegyes egész bilineáris programozási (*mixed integer bilinear programming*, MIBLP) modellel írták le. A modellt linearizálták úgy, hogy egy lineáris feltételekkel korlátozott MIBLP feladatot kapjanak, amit egy módosított külső közelítési (*outer approximation*) algoritmussal oldottak meg. A globális optimumot nem tudták garantálni és gyenge futási eredményeket értek el egy szakirodalmi példa esetében is.

Ierapetritou és Floudas (1998) az STN leírást és az eseménypontokat használó folytonos idejű modellen alapuló módszert készített. Egy eseménypont reprezentálja egy taszk elkezdését vagy befejezését, a rendelkezésre álló időintervallum egy tetszőlegesen választott időpillanatában, ahol az eseménypontok optimális számát egy iteratív módszerrel határozzák meg. Az előző formalizmustól való különbség abban van, hogy ebben a megközelítésben az eseménypontok nem csak egy időpontot azonosítanak, hanem két eseménypont közötti távolságot is ezzel definiálják. A modell fő erénye az eredő MILP feladat méretének nagy mértékű csökkentése volt, amit úgy értek el, hogy a taszkokhoz és a berendezésekhez rendelt eseményeket szétválasztották, a berendezések nem a

taszkokhoz vannak rendelve, hanem az eseménypontokhoz. A módszer fő hátránya, hogy nem tudja az optimumot biztosítani.

Graells és társai (1998) bevezették az esemény-művelet hálózat (*event-operation network*, EON) leírást, amely egyszerűsítette az időzítés részfeladatát. Az EON leírás a folyamat-anyag hálózat (*process-material network*, PMN) leírással együtt használva robosztus keretet ad a részletes folyamat szimulációhoz az ütemezés szintjén.

A disszertációban az S-gráf leírást használjuk, amelynek alapjait Sanmartí és társai (1998) fektették le. Az S-gráf egy irányított konjuktív gráf, ahol a csomópontok jelölik a taszkokat és az élek a taszkok sorrendjét. Ehhez a leíráshoz Sanmartí és társai (2002) készítettek egy speciális szétválasztás és korlátozás algoritmust, amelynek segítségével az optimális megoldáshoz tartozó S-gráf megtalálható.

3. Feladat megfogalmazás

Egy többcélú ütemezési feladat megadásához, ahol a termékek előállítási módja különböző lehet, háromféle információra van szükség. Ezek a termékek receptjei, a taszkokhoz rendelhető berendezések és a szükséges termékmennyiségek. A recept tartalmazza azokat a minimális információkat, melyek egy termék előállításához szükségesek. Az ISA SP88 szabvány négy szintjét definiálja a recepteknek, melyek használata a felhasználás szintjétől függ. A szintek a következők:

- Az általános recept (*general recipe*) azonosítja a nyersanyagokat, a relatív mennyiségüket, és az előállítandó anyagmennyiséget. Ez a recept nem tartalmaz berendezés és hely (például gyár) specifikus információkat.
- A hely recept (*site recipe*) hely specifikus információkkal kibővített általános recept.
- A mester recept (*master recipe*) tartalmazza a berendezés specifikus információkat (például működési idő), a nyersanyagokat a rendelkezésre álló mennyiségekkel, valamint a termékek előállításának folyamatát.
- A kontrol recept (*control recipe*) a mester recepthez képest tartalmaz további információkat azon berendezésekről, melyeket egy termék egy batch-jének előállítására használnak.

A négy recepttípus közül a gyakorlatban a mester receptet használják szakaszos folyamatok ütemezési feladatának leírásához. A továbbiakban mi is ezt használjuk és egyszerűen receptnek nevezzük.

1.példa

Tekintsük a következő ütemezési feladatot, amelyhez tartozó termékek előállítási módjai (receptjei) és a felhasználható berendezések az 1. táblázatban láthatóak. A szükséges termékmennyiségek és a nyersanyagok mennyiségei implicit módon a batch-ek számával adottak a 2. táblázatban.

1. táblázat: Recept az 1. példához

Taszk	A termék		B termék		C termék	
	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)
1	E1	6	E3	9	E2	7
2	E2	9	E3	15	E1	17
3	E1	14	E2	16	E3	8

2. táblázat: Batchek-ek száma az 1. példához.

Termék	A	B	C
Batch-ek száma	1	1	1

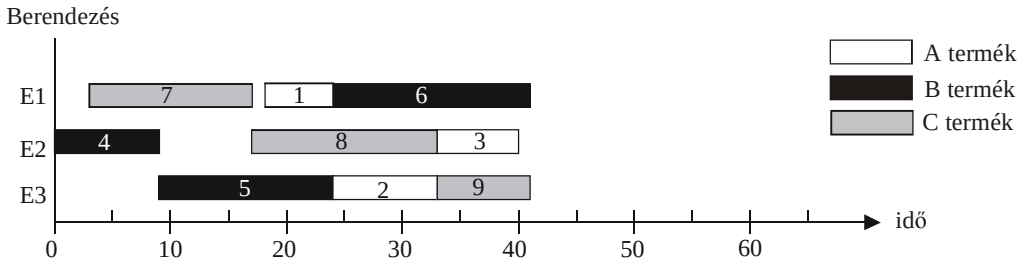
A feladat egyértelmű leírásához meg kell határozni, hogy a köztes termékeket a taszkok között el lehet-e tárolni például egy tároló berendezésben. Ha lehet, akkor milyen mennyiségben lehet tárolni az anyagot és a tárolókat hol (mely taszkok vagy berendezések között) lehet használni; ha nem, akkor a gyártó berendezésben várakozhat-e az anyag. Ezen tulajdonságok alapján a következő tárolási stratégiákat különböztethetjük meg:

- UIS tárolási stratégia (*unlimited intermediate storage policy*): a köztes termékeket végtelen mennyiségben lehet tárolni. A taszk elvégzése után a berendezésből a köztes terméket tároljuk egy „végtelen” kapacitású tárolóban, azaz a berendezést tisztítás után rögtön tudjuk használni.
- NIS tárolási stratégia (*non intermediate storage policy*): a köztes termékek tárolására a taszkok között nincs lehetőség. A taszk elvégzése után a köztes terméket a berendezésben tárolhatjuk, azaz a berendezés akkor áll rendelkezésre, ha az anyagot áttöltöttük egy másik, értelemszerűen a következő taszkot végrehajtó berendezésbe.
- FIS tárolási stratégia (*finite intermediate storage policy*): a rendszer véges számú és méretű tárolót tartalmaz, egy tároló csak két előre meghatározott berendezés között használható.
- ZW tárolási stratégia (*zero wait policy*): a köztes terméket nem tárolhatjuk sem az őt előállító berendezésben, sem tároló berendezésekben, azonnal át kell tölteni a következő berendezésbe.

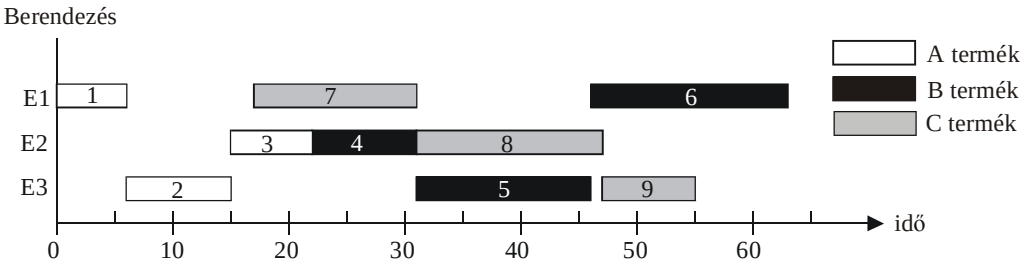
- MIS tárolási stratégia (*mixed intermediate storage policy*): az előző négy tárolási stratégia keveréke, a rendszer különböző pontjain különböző stratégiák lehetnek.
- CIS tárolási stratégia (*common intermediate storage policy*): a rendszer véges számú és méretű tárolót tartalmaz. Ez a tárolási stratégia abban különbözik a FIS stratégiától, hogy itt a tárolók helye nem rögzített.

A szakirodalomban túlnyomórészt az UIS stratégiával foglalkoztak, amely stratégia főleg a gépiparban használatos, ahol például egy raktárhelységgel megoldható nagy mennyiségű köztes termék tárolása is. Gyakorlatban viszont figyelembe kell vennünk azokat az eseteket is, amikor nincs lehetőség köztes tárolásra (NIS stratégia). Például vegyipari rendszerek ütemezésnél, amikor folyékony és néha instabil köztes termékek szerepelnek a rendszerben, akkor ezeket nem lehet tárolni.

A tárolási stratégiától függ a megvalósítható (*feasible*) megoldások halmaza. Ha például egy ütemezés UIS esetben megvalósítható, akkor nem biztos, hogy NIS esetben is az. Az 1. ábrán Gannt diagram mutatja az 1. példa optimális ütemezését UIS esetben, amely ütemezés NIS esetben nem megvalósítható. Az ábrán látható, hogy az E1 berendezésből (1. taszk) az anyag az E3 berendezésbe (2. taszk) ugyanakkor töltődik át, amikor az E3-ból (5. taszk) az E1-be (6. taszk). Ez az anyagáramlás akkor lenne megvalósítható, ha rendelkezésre állna egy köztes tároló, ahol az egyik anyagot tárolni lehetne, amíg a másik áttöltődik, ez NIS esetben nem lehetséges. A feladat optimális megoldása NIS esetén a 2. ábrán látható.



1. ábra: Az 1. példához tartozó optimális megoldás UIS esetben Gannt diagram segítségével.



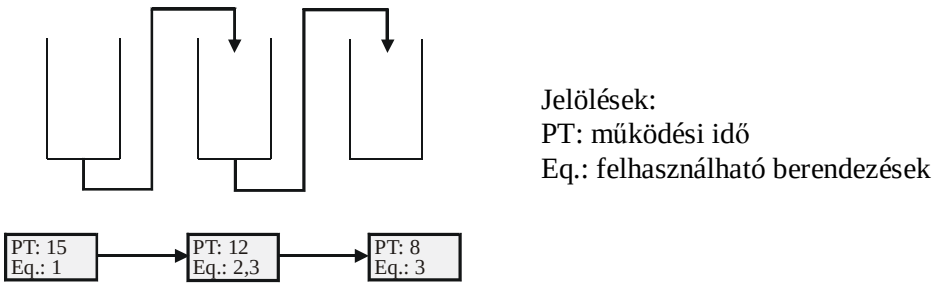
2. ábra: Az 1. példához tartozó optimális megoldás NIS esetben Gannt diagram segítségével.

Egy egyszerű recept leírásához, amikor a termékek előállítása egy vonalon történik, elegendőek a fentiekben bemutatott információk. Összetett recept esetén, amikor a receptben van több kimenettel, illetve több bemenettel rendelkező taszk, szükségünk lehet további adatokra a feladat korrekt leírásához. Amikor egy taszknak több bemenete is van, akkor a bemenetek időzítése is fontos. Lehet, hogy a bemeneteknek egyszerre jelen kell lenniük a taszk kezdeténél, a bemenetek sorrendje rögzített, a bemenetek tetszőleges sorrendben rendelkezésre állhatnak, illetve lehetséges ezen esetek kombinációja is. Ez az időzítés a megvalósíthatóságra is hatással lehet, például az első esetben (egyidejű bemenetek esetén) NIS stratégiát használva egy taszk két bemenetét nem állíthatja elő ugyanaz a berendezés, a többi esetben viszont igen.

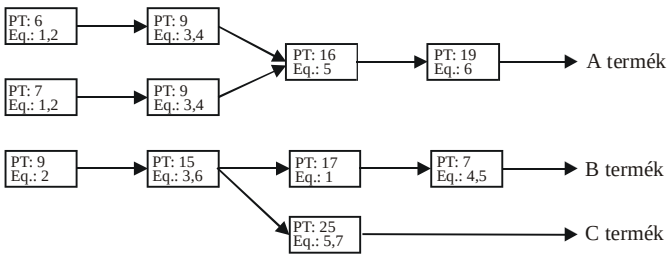
A következő fejezetben bemutatunk egy gráf leírást (S-gráf) többcélú ütemezési feladatokhoz, amely a feladat összes eddig bemutatott strukturális tulajdonságát jól ábrázolja. A leírás NIS tárolási stratégiához készült, de bármely fent említett tárolási stratégiához alkalmazható.

4. S-gráf leírás az ütemezés strukturális ábrázolásához

A termékek előállítását leíró receptet hagyományosan lehet ábrázolni egy irányított gráffal, ahol a gráf csúcsai jelentik a taszkokat, a közöttük lévő élek pedig ezek sorrendjét. A működési idők és a taszkokhoz felhasználható berendezések a megfelelő csúcsokban adottak. A 3. ábra mutatja egy három lépésben (taszk) előállítható termék hagyományosan megadott receptjét. Természetesen összetett receptek is leírhatóak ezen a módon, például a 4. ábrán látható receptben az A terméket két köztes anyag összekeverésével, majd a keverék további feldolgozásával lehet előállítani.



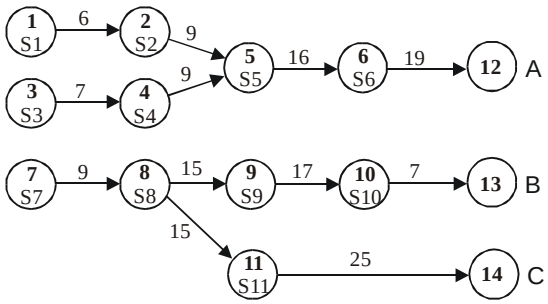
3. ábra: Három lépésben (taszkkal) előállítható termék receptjének hagyományos leírása.



4. ábra: Három termék előállításának hagyományos leírása.

Ebben a hagyományos leírásban az élek a taszkok sorrendjét adják meg, minden egyéb információ a csomópontokhoz van rendelve. A 4. ábrán látható recept gráf leírása az 5. ábrán látható, ahol S*i* azoknak a berendezéseknek a halmaza, amelyekkel az *i* csomóponttal reprezentált taszkot végre lehet hajtani (például S1={E1, E2}), valamint egy-egy további csomópont jelöl minden terméket. Ebben a leírásban az élek súlya alsó korlátot jelent a két kapcsolódó

taszk kezdési idejének különbségére. Egy taszk működési ideje a hozzá rendelhető berendezésektől függően különböző lehet, ebben az esetben az él súlya a felhasználható berendezések működési idői közül a legkisebb lesz. Ez az érték a feladat megoldása során természetesen változhat.



5. ábra: Gráf leírás a 4. ábrán látható recepthez.

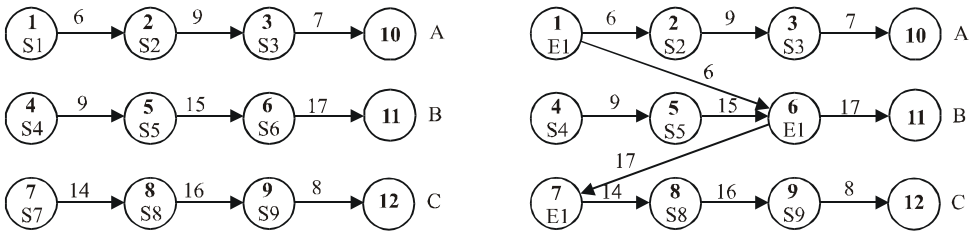
Ha egy szakaszos működésű gyártási folyamat struktúrája kört tartalmaz, az nem eredményez kört a receptet leíró gráfban, mivel a látszólagos recirkuláció egy időben későbbi batch valamely taszkjának a betáplálását jelenti. Ezért feltételezhetjük, hogy bármely recept egy körmentes irányított gráffal ábrázolható.

4.1. NIS és UIS tárolási stratégiák

A hagyományosan használt megoldó eszközök a gépipar általános ütemezési feladatainak megoldására korlátozottak, ahol a köztes termékeket két taszk között gyakorlatilag végtelen mennyiségben tárolni lehet. Gyakorlatban viszont egy másik fontos esetet is figyelembe kell vennünk, amikor nincs lehetőség köztes tárolásra, ekkor kifinomultabb gráf leírásra és algoritmusokra van szükség.

Tegyük fel, hogy adott minden termékre a legyártandó mennyiség és adott, hogy melyik taszk melyik berendezéssel vagy berendezésekkel hajtható végre. A taszkok sorrendje ábrázolható egy irányított diszjunktív gráffal (Adams és társai, 1988), ahol a taszkok sorrendje élekkel adott úgy, hogy az élek kötik össze az egy berendezéshez tartozó taszkokat jelölő csomópontokat. Az élek a taszkok működési idejével súlyozottak és megadják ezek működési sorrendjét, figyelembe véve a receptet is. Az 1. példa receptje a 6. (a) ábrán látható, ahol az E1 készülék berendezés 1-6-7 működési sorrendjéhez tartozó hagyományos gráf leírása a 6. (b)

ábrán látható. Ebben a példában az E1 berendezés az 1., a 6. és a 7. taszkokhoz van rendelve, ahol E1∈ S1, S6, S7. Megfigyelhető a 6. (b) ábrán, hogy a 6. taszk nem kezdődhet korábban, mint ahogy az 5. taszk (recept) és az 1. taszk (működési sorrend) befejeződött.



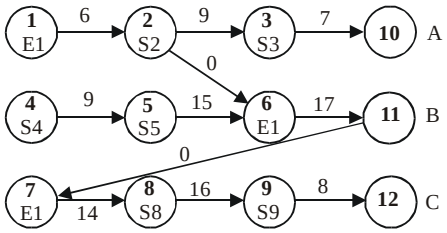
(a) recept (b) Az E1 berendezés működési sorrendje

6. ábra: Az E1 berendezés 1-6-7 működési sorrendjének megadása UIS esetén.

Ez a fajta leírás jól használható UIS típusú feladatok megoldására, de nem megfelelő NIS esetén. Az UIS feltételezi, hogy a berendezések a taszkok végeztével azonnal rendelkezésre állnak, azaz a köztes termékek, amelyek a taszk végrehajtása során keletkeznek, kikerülnek a berendezésből és eltárolásra kerülnek, amíg a következő taszk el nem kezdődik. A legtöbb szakaszos működésű folyamatban ezt nem lehet biztosítani, azaz NIS esetet kell figyelembe vennünk.

NIS esetben a berendezések nem állnak rendelkezésre egy taszk befejezése után egészen addig, amíg a tárolt anyag át nem töltődik a következő taszkot végrehajtó berendezésbe. A gráfban ezt a további feltételt is reprezentálni kell illetve lehet egy vagy több él segítségével a következő módon. Jelölje t_j azokhoz a taszkokhoz tartozó csomópontokat, amelyek a recept szerint a *j* csomóponthoz tartozó taszkot követik. Ha az E*i* berendezés a *k* taszkot végzi el közvetlenül a *j* után, akkor egy nulla súlyú (vagy váltási idővel súlyozott) élet húzunk t_j minden eleméből *k*-ba. Ezt a fajta leírást nevezzük S-gráfnak. A 7. ábrán látható az E1 berendezés 1-6-7 működési sorrendjének megadása S-gráf segítségével. Ahelyett, hogy az UIS esetben szokásos módon kötnénk össze az 1. és a 6. valamint a 6. és a 7. csomópontot működési időkkel súlyozott élekkel (ahogy az a 6. (b) ábrán

látható), nulla súlyú élek vannak a 2. és a 6. valamint a 11. és a 7. csomópontok között.



7. ábra: Az E1 berendezés 1-6-7 működési sorrendjének megadása NIS esetén.

A hagyományos leírás nem alkalmazható NIS esetre, de az S-gráf mind NIS mind UIS esetben jól használható ütemezési feladatok leírására (például további csomópontok bevezetésével a tárolási lehetőségek jelölésére és tároló berendezések felvételével a köztes tárolók ütemezéséhez). Továbbá az S-gráf leírás alkalmas szakaszos és folyamatos működésű berendezéseket is tartalmazó rendszerek leírására is.

Egy irányított G gráfot egy (N, A) párral adhatunk meg, ahol N véges halmaz, a csomópontok halmaza és A az élek halmaza ($A \subseteq N \times N$). Egy S-gráfnak két fajta éle van (léteznek úgynevezett recept-élek és ütemezési-élek), így egy S-gráfot $G(N, A_1, A_2)$ formában adhatunk meg, ahol N a csomópontok, A_1 a recept-élek, A_2 pedig az ütemezési-élek halmaza feltéve, hogy $A_1 \subseteq N \times N$, $A_2 \subseteq N \times N$ és $A_1 \cap A_2 = \varnothing$; továbbá minden $(i, j) \in A_1 \cup A_2$ élhez tartozik egy nem negatív érték $c(i, j)$, az él súlya. Ha egy él i-ből j-be mutat ($(i, j) \in A_1 \cup A_2$), akkor az a gyakorlatban azt jelenti, hogy az j csomóponthoz tartozó taszk legalább $c(i, j)$ idővel később kezdi a működését, mint az i csomóponthoz tartozó taszk. Speciális S-gráf tartozik a recepthez (recept-gráf) illetve a megoldáshoz (ütemezési-gráf).

4.2. Recept-gráf

Egy ütemezési feladat receptje megadja minden termékhez a termékekhez tartozó taszkok sorrendjét, a köztük lévő anyagáramokat, valamint az egyes taszkokhoz

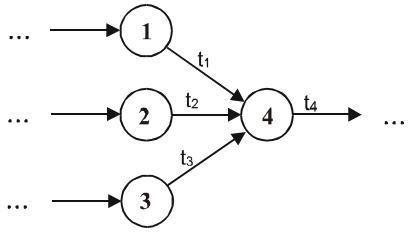
felhasználható berendezések halmazát. Ezeknek az információknak a recept-gráfban szerepelniük kell ahhoz, hogy a receptet megfelelően le tudjuk vele írni.

Rendeljünk egy-egy csomópontot minden taszkhoz (taszk-csomópont) és egy-egy csomópontot minden termékhez (termék-csomópont). Egy él (recept-él) mutasson minden taszk-csomópontból a receptben utána következőbe, valamint a terméket gyártó csomópontokból a megfelelő termék-csomópontba. Egy él súlya legyen az él kezdő csomópontjához tartozó taszk működési idejével egyenlő abban az esetben, ha a taszkot csak egy berendezés hajthatja végre. Ha több berendezés is rendelkezésre áll, akkor a működési idők közül a legkisebbel egyezzen meg az él súlya.

Ha egy termékből nagyobb mennyiségre van szükség, mint amennyit a recept alapján egyszerre elő lehet állítani (több batch-re van szükség), akkor a termék előállításában résztvevő taszkok taszk-csomópontjai, a termék-csomópont, valamint a közöttük lévő élek többször bekerülnek a gráfba. Az így keletkezett gráfot hívjuk taszk-hálózatnak, ahol N_t jelöli a taszk-csomópontok, N_p pedig a termék-csomópontok halmazát ($N_t \cap N_p = \varnothing$).

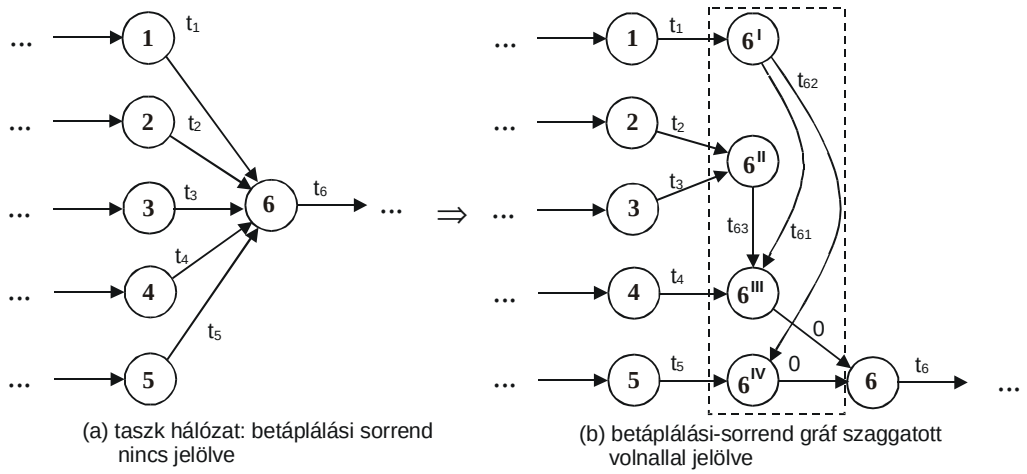
Fontos megjegyezni, hogy több batch esetén egy taszkhoz több taszk-csomópont is tartozik. Ha a feladat szempontjából fontos, hogy egy taszkhoz több csomópont is tartozik, akkor ezen csomópontok által reprezentált termelési folyamatot (és továbbiakban magát a csomópontot is) tevékenységnek nevezzük, egyébként a taszk kifejezést használjuk.

A gyakorlatban egy taszk bemeneteinek nem kell feltétlenül egyszerre táplálniuk a taszkot, hanem létezhetnek további feltételek a bemenetek időzítésével kapcsolatban. Formálisan a bemenetek időzítését egy részleges rendezéssel (*partial order*) adhatjuk meg, amit egy gráffal (betáplálási-sorrend gráf) lehet reprezentálni. A legegyszerűbb eset, amikor a bemeneteknek egyszerre kell rendelkezésre állniuk. A 8. ábrán a 4. taszknak három egymástól független bemente van (1., 2. és 3. taszk) amelyek párhuzamosan működnek. Nyilvánvalóan mind a három berendezésnek egyszerre jelen kell lennie a 4. taszk kezdeténél.



8. ábra: Recept-gráf része: egyidejű bemenetek egy taszkhoz.

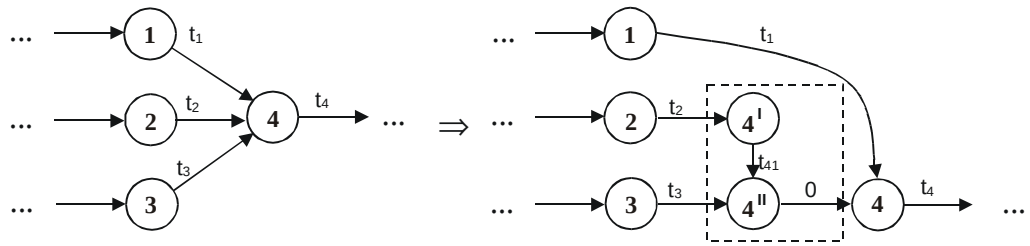
Tegyük fel, hogy egy taszknak több bemenete van. Ilyenkor bármely két bemenet között a következő kapcsolatok állhatnak fenn: a bemeneteknek egyszerre kell rendelkezésre állniuk, sorrendjük rögzített, vagy sorrendjük tetszőleges. A bemenetek közötti kapcsolatrendszert egy részleges rendezés írja le, ezt a rendezést egy a taszk-hálózatba beillesztett körmentes irányított gráffal reprezentálhatjuk. Tegyük fel, hogy egy taszknak öt bemenete van (9. (a) ábra), az 1. bemenet megelőzi a 4. és az 5. bemenetet, a 2. bemenet megelőzi a 4. bemenetet, valamint a 2. bemenetnek a 3. bemenettel egyszerre kell rendelkezésre állnia. A betáplálási-sorrend gráffal kiegészített taszk hálózat a 9. (b) ábrán látható. A betáplálási-sorrend gráf csomópontjait betáplálási-sorrend csomópontoknak hívjuk (halmazukat N_f -el jelöljük), élei recept-élek.



9. ábra: Recept-gráf részlet: betáplálási-sorrend a 6. taszkhoz.

Külön meg kell említenünk a folyamatos betáplálás esetét (*fed-batch operation*), amikor egy bemenetnek a taszk végrehajtása közben folyamatosan jelen kell lennie. Ebben az esetben a recept-él a bemenet csomópontjából

közvetlenül a taszk csomópontba mutat, kikerülve az esetleges betáplálási-sorrend gráfot (10. ábra). Ebben az esetben a bemenetet előállító berendezés működési sorrendjét jelző él súlya megnövekszik a taszk végrehajtási idejével.



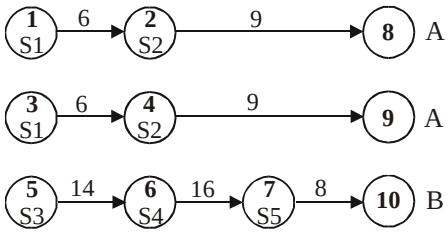
(a) taszk hálózat (b) az 1. bemenet folyamatosan rendelkezésre áll

10. ábra: Folyamatos betáplálás.

A taszk-hálózat kiegészítve a szükséges betáplálási-sorrend gráfokkal alkotja a recept-gráfot. (A könnyebb érthetőség és az egyszerűbb formalizmus kedvéért a továbbiakban feltételezzük, hogy a recept-gráfban nincsenek betáplálási-sorrend gráfok, azaz $N_f = \varnothing$.) A recept-gráfban minden a receptben található strukturális információ adott. Ha $G(N, A_1, A_2)$ egy recept-gráf, akkor körmentes és $A_2 = \varnothing$. Legyen $N_i (\subset N_t)$ azon csomópontok halmaza, amelyek az i berendezéssel végrehajthatók. Feltehetjük, hogy minden taszkhoz létezik legalább egy berendezés, amivel végre lehet hajtani, azaz a taszk-csomópontok halmaza megadható a következő módon $N_t = N_1 \cup N_2 \cup \mathbf{L} \cup N_n$, ahol N_i és N_j $(i, j = 1,2,...,n)$ tartalmazhat azonos elemeket, azaz metszetük nem szükségszerűen üres.

2. példa

Tegyük fel, hogy két terméket kell előállítanunk, A-t és B-t, A-ból két batch-nyit, B-ből egyet. A-t két egymás utáni lépésben lehet előállítani, ahol az első lépés az S1, a második pedig az S2 halmazban lévő berendezések bármelyikével végrehajtható 6 illetve 9 időegység alatt. B-t három egymást követő lépésben lehet előállítani az S3, az S4 illetve az S5 halmazokban lévő berendezésekkel 14, 16 illetve 8 időegység alatt. A feladat recept-gráfja a 11. ábrán látható.



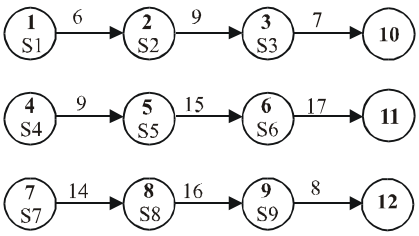
11. ábra: A 2. példa recept-gráfja: két batch az A termékből és egy a B-ből.

4.3. Ütemezési-gráf

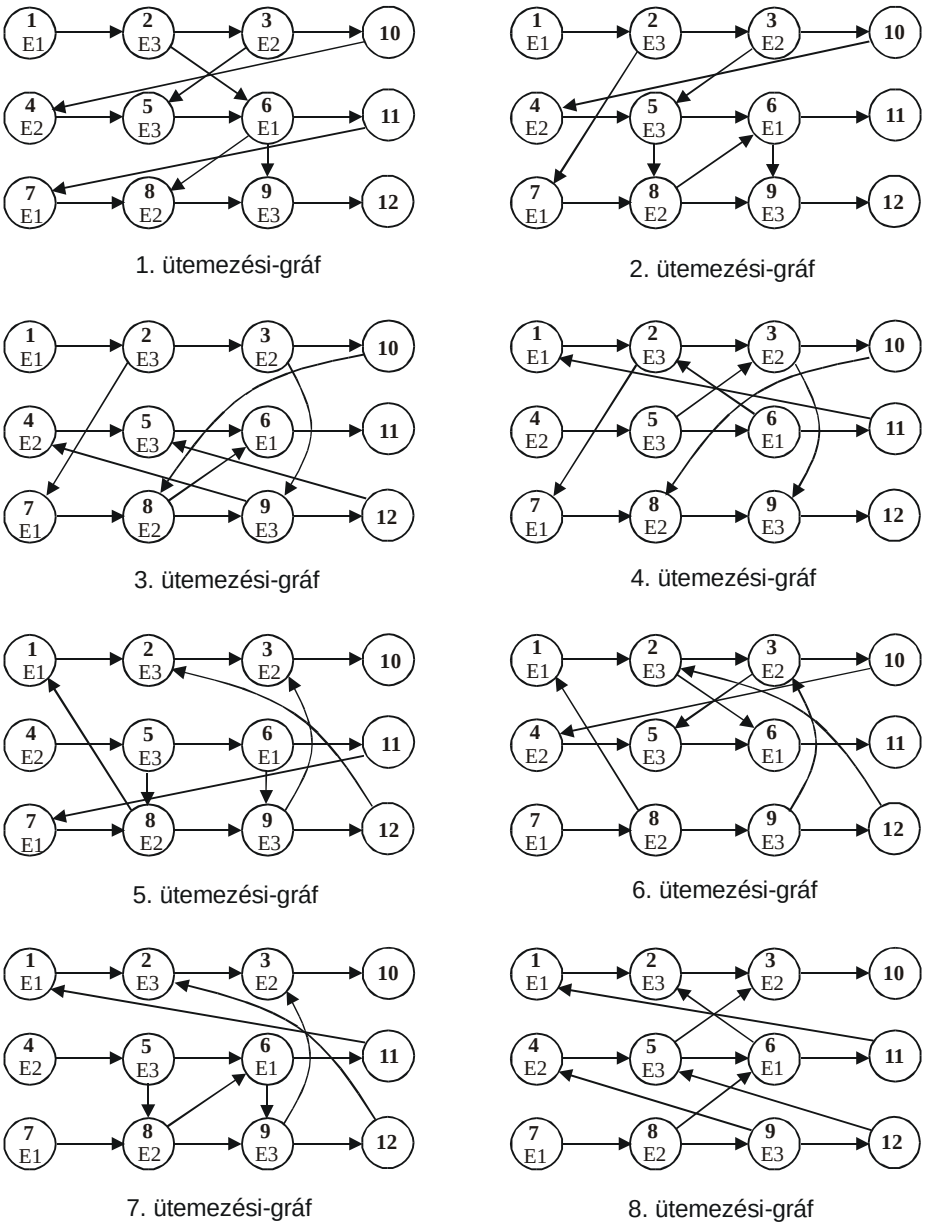
Az ütemezési-gráf olyan speciális S-gráf, amely egy megoldást reprezentál; az ütemezési feladat minden megoldásához létezik egy ütemezési-gráf. A $G'(N, A_1, A_2)$ S-gráfot a $G(N, A_1, \varnothing)$ recept-gráfhoz tartozó ütemezési-gráfnak nevezünk, ha a recept-gráf minden csomópontja (taszkja) ütemezve van figyelembe véve a berendezés-taszk hozzárendelést. Egy megfelelő keresési módszerrel az optimális megoldáshoz tartozó ütemezési-gráf és berendezés-taszk hozzárendelés megtalálható.

3. példa

Három termék előállítása a 12. ábrán látható recept-gráf szerint történik. Minden taszkhoz egy berendezés áll rendelkezésre, ahol S1={E1}, S2={E3}, S3={E2}, S4={E2}, S5={E3}, S6={E1}, S7={E1}, S8={E2}, S9={E3}. A feladatnak nyolc különböző megoldása létezik, amelyek ütemezési-gráfjai a 13. ábrán láthatóak.



12. ábra: Recept-gráf a 3. példához.



13. ábra: A 3. példa ütemezési-gráfjai.

Az ütemezési-gráf formális leíráshoz tételezzük fel, hogy adott a $G(N, A_1, \varnothing)$ recept-gráfhoz tartozó $G'(N, A_1, A_2)$ ütemezési-gráf, ahol $A_2 \subseteq N \times N$. Jelölje M_i $(i = 1,2,...,n)$ azoknak a csomópontoknak a halmazát, amelyekhez tartozó taszkokat az i berendezés hajtja végre. Feltételezzük, hogy a megoldásban egy taszkot pontosan egy berendezéssel lehet végrehajtani, azaz $M_i \cap M_j = \varnothing$ $(i \neq j,\ i, j = 1,2,...,n)$. Ebből következik, hogy $M_1, M_2, ..., M_n$ egy partícionálása a taszk csomópontok halmazának ($N_t = N_1 \cup N_2 \cup \mathbf{L} \cup N_n$) úgy, hogy $M_i \subseteq N_i$ $(i = 1,2,...,n)$.

Ahhoz, hogy az ütemezési-gráf leírását meg tudjunk adni, szükségünk van egy speciális S-gráf bevezetésére, amelyet komponens-gráfnak nevezünk. A komponens-gráfok mutatják meg az egyes berendezések működési sorrendjét. Formálisan az i berendezéshez tartozó komponens-gráf $G_i'(N_i', A_{1i}, A_{2i})$ ($\subseteq G'(N, A_1, A_2)$) S-gráf ($i = 1,2,...,n$), ahol

- N_i' tartalmazza G'-ből az M_i összes csomópontját és az összes olyan csomópontot, amelybe M_i-ből induló recept-él mutat

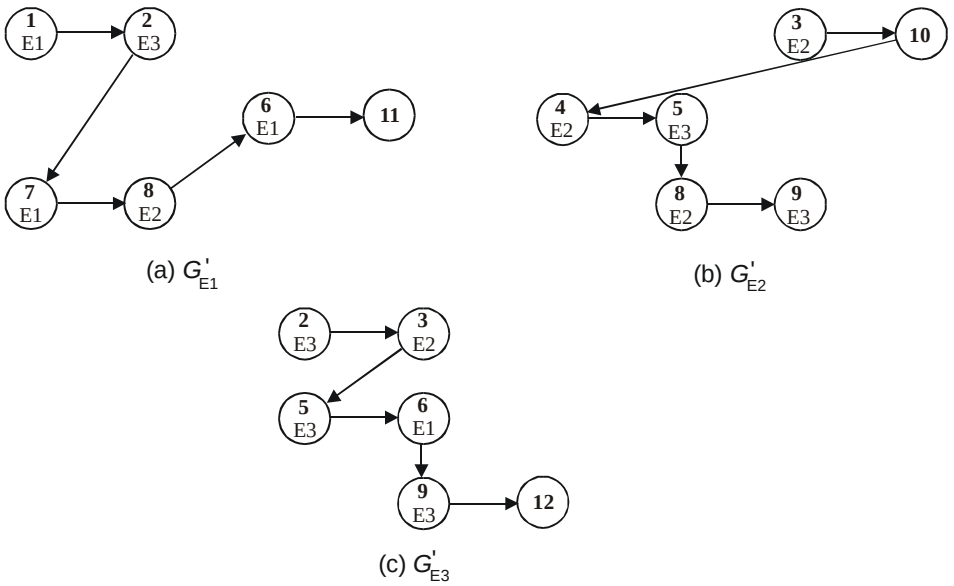
$$N_i' = M_i \cup \{k : k \in N \text{ és } \exists j \in M_i, \text{hogy } (j,k) \in A_1\}$$

- A_{1i} tartalmazza az összes olyan recept-élet G'-ből, amely M_i-ből indul

$$A_{1i} = \{(j,k) : (j,k) \in A_1 \text{ és } j \in M_i\})$$

- A_{2i} tartalmazza az összes ütemezési-élet G'-ből, amely A_{1i} valamely élének végpontjából M_i valamelyik elemébe mutat

$$A_{2i} = \{(j,k) : (j,k) \in A_2, k \in M_i \text{ és } \exists l \in M_i, \text{hogy } (l,j) \in A_{1i}\}$$



14. ábra: A 3. példa 2. ütemezési-gráfjának (13. ábra) komponens-gráfjai.

3. példa (további vizsgálat)

A 2. ütemezési-gráf (13. ábra) komponens-gráfjai G_{Ei}' ($i = 1,2,3$) a 14. ábrán láthatóak a következő berendezés-taszk hozzárendeléssel: $M_{E1} = \{1,6,7\}$, $M_{E2} = \{3,4,8\}$, $M_{E3} = \{2,5,9\}$.

A komponens-gráf segítségével a következőképpen definiáljuk az ütemezési-gráfot. A $G(N, A_1, \varnothing)$ recept-gráfhoz és az M_i $(i = 1,2,...,n)$ berendezés-taszk hozzárendeléshez tartozó $G'(N, A_1, A_2)$ S-gráf ütemezési-gráf, ha teljesíti a következő négy feltételt.

(SG1)

G' nem tartalmaz irányított kört.

(SG2)

G_i' komponens gráf által meghatározott rendezés teljes minden $M_i \cup \{j\}$ halmazon, ahol $j \in N_i'$ és $i = 1,2,...,n$.

(SG3)

Az N_i' $(i = 1,2,...,n)$ halmazok minden eleméből legfeljebb egy A_{2i}-beli él indul.

(SG4)

A G' ütemezési-gráf megegyezik komponens-gráfjainak uniójával, azaz $G' = \bigcup_{i=1}^{n} G_i'$.

Az ütemezési-gráfok és a megvalósítható ütemezések közötti kapcsolat azon alapul, hogy az (i, j) él az S-gráfban azt mutatja, hogy a j csomópont által reprezentált taszk leghamarabb $c(i, j)$ idővel később kezdődhet el, mint az i által reprezentált. Ebből következik, hogy ha az (SG1) feltétel nem teljesül, akkor a megoldás nem megvalósítható, vagy az időbeliség vagy a *NIS* feltételei nem teljesülnek. Ha (SG1) feltétel teljesül, de (SG2) nem, akkor az ütemezés nem teljes, mivel létezik olyan berendezés mellyel végrehajtandó taszkok sorrendje nem egyértelműen meghatározott. Bár (SG3) és (SG4) feltételek teljesülése nem szükséges a megvalósíthatósághoz és a teljességhez, de az optimális megoldás mindig megtalálható az (SG3) és (SG4) által leszűkített keresési térben. Az (SG3) biztosítja, hogy ne legyenek redundáns-ütemezési élek a gráfban, az (SG4) feltétel pedig kiszűri azokat az éleket, amelyek nem tartoznak egyik berendezés

ütemezéséhez sem. Következésképpen az optimális megoldás mindig megadható az ütemezési-gráfok segítségével. Az (SG3) és (SG4) feltételek teljesülése esetén az S-gráf minimális abban az értelemben, hogy él elhagyásával nem ír le megoldást.

5. Alapalgoritmus az S-gráffal leírt ütemezési feladatok megoldásához

A fejezetben Sanmartí és társai (2002) által bevezetett alapalgoritmust mutatjuk be, amely általánosan használható bármilyen típusú ütemezési feladatra, ahol az algoritmus bemenete a feladat recept-gráfja, kimenete pedig egy optimális ütemezés ütemezési-gráfja. Az alapalgoritmust kiterjesztettem egy speciális feladatosztályra (6. fejezetben) illetve kiegészítettem általános célú vagy feladat specifikus gyorsítási eljárásokkal (7. fejezetben).

Az algoritmusban a cél az optimális megoldás (ütemezési-gráf) megtalálása a receptből (recept-gráf) kiindulva részfeladatok (általános S-gráf) sorozatán keresztül. A recept-gráf mindig részgráfja a hozzá tartozó összes ütemezési-gráfnak úgy, hogy a gráfok csomópontjai nem változnak, és a recept-éleknek is legfeljebb a súlya növekedhet. Egy ütemezési-gráfnak minden éle, amely nem tartozik a kiinduló recept-gráfhoz, ütemezési-él. A recept-gráf összes lehetséges kiterjesztése ütemezési-élekkel (figyelembe véve az (SG1)-(SG4) feltételeket és a berendezés-taszk hozzárendelési lehetőségeket) elvezet bennünket az összes ütemezési-gráfhoz. Mivel a lehetséges kiterjesztések száma véges, az összes ütemezési-gráf a hozzá tartozó berendezés-taszk hozzárendelésekkel véges lépésben előállítható. Ez a gráf leírás jó alap egy szétválasztás és korlátozás (*branch-and-bound*, B&B) típusú algoritmushoz.

5.1. Az algoritmus szétválasztási lépése

A javasolt szétválasztás és korlátozás algoritmus képes előállítani bármely recept-gráfhoz a hozzá tartozó összes ütemezési-gráfot, ahol minden részfeladathoz egy S-gráf és egy részleges berendezés-taszk hozzárendelés tartozik, valamint természetesen egy csomópont a keresőfában. A recept-gráf (berendezés-taszk hozzárendelés nélkül) a gyökere a keresőfának. Minden részfeladatnál kiválasztunk egy berendezést. A részfeladat minden gyerek részfeladatában a berendezést hozzárendeljük egy megfelelő be nem ütemezett taszkhoz (természetesen mindegyikben másikhoz), valamint a kiválasztott taszkot

beütemezzük a berendezés aktuálisan utolsó lépésének. Mivel a taszkok működési ideje függhet a felhasznált berendezéstől, ezért egy berendezés hozzárendelése egy taszkhoz módosíthatja a taszkot jelölő csomópontból kiinduló recept-él illetve recept-élek súlyát. Az egyszerűség kedvéért feltételeztük, hogy a megoldásban egy taszkot pontosan egy berendezés hajthat végre. A ***main*** eljárás állítja be a változók kezdő értékeit (15. ábra) majd meghívja a ***szétválasztás*** eljárást (16. ábra).

```
procedure main
jelölések:  n: berendezések száma
            N_i (i = 1,2,..., n): az i berendezéssel végrehajtható taszkok halmaza
            last_node: (i, j) párok halmaza, ahol i egy berendezés és j egy taszk
            PP = (G(N, A_1, A_2), bound, last_node, SOUN): részfeladat
bemenet: recept-gráf G(N,A_1,∅) és N_i (i = 1,2,...,n)
begin
    SET = ∅;
    bound = 0;
    SOUN = N_1∪N_2∪...∪N_n;
    last_node = ∅;
    current_best = ∞;
    tegyük (G(N, A_1, ∅), bound, last_node, SOUN)-t a SET halmazba;
    while SET ≠ ∅ do
        vegyünk ki egy elemet a SET-ből, jelöljük PP-vel;
        szétválasztás(PP);
    if current_best < ∞ then
        print solution;
end
```

15. ábra: Az ütemezési algoritmus *main* eljárása.

Az algoritmus megvalósításánál nagy szabadságot ad a keresési stratégia kiválasztása. Például az ütemezendő berendezés kiválasztása (***szétválasztás*** eljárás) nagyban befolyásolhatja az algoritmus hatékonyságát. Gyakorlatban a berendezések „leterheltségének” sorrendjében való ütemezése jó stratégiának bizonyult. Ezen kívül a részfeladat kiválasztása (***main*** eljárás) is jelentős hatással van az algoritmus futási idejére. A számítógépes megvalósításnál mélységben először keresést alkalmaztunk, az egy szinten lévő részfeladatok közül a legjobb alsó korlátút választva.

```
procedure szétválasztás(PP)
megjegyzés: PP részfeladat összes gyerek részfeladatának generálása
jelölések:  graph(PP) : a PP részfeladat S-gráfja
            bound(PP) : a PP részfeladat alsó korlátja
            last_node(PP) : a PP részfeladathoz tartozó párok halmaza
            SOUN(PP) : a PP részfeladat SOUN halmaza
begin
    legyen EQ egy berendezés amelyre N_EQ ∩ SOUN(PP) ≠ ∅;
    SO = N_EQ ∩ SOUN(PP);
    for all k ∈ SO do
        G0(N, A1, A2) = graph(PP);
        for all (k, l) ∈ A1 do
            c(k, l) módosítása;
        if ∃(i, j) ∈ last_node úgy, hogy i = EQ then
            for all (j, l) ∈ A1 do
                G0(N, A1, A2) = G0(N, A1, A2∪{(l, k)});
        korlátozás(G0(N, A1∪A2), bound);
        else
            if bound < current_best then
                if SOUN(PP)\k = ∅ then
                    current_best, SET és solution módosítása;
                else if ∃(i, j) ∈ last_node úgy, hogy i = EQ then
                    tegyük (G0(N, A1, A2), bound,
                      last_node(PP)∪{(EQ, k)}\{(EQ, j)}, SOUN(PP)\{k})-t
                      SET-be;
                else
                    tegyük (G0(N, A1, A2), bound, last_node(PP)∪{(EQ, k)},
                      SOUN(PP)\{k})-t SET-be;
end
```

16. ábra: Az ütemezési algoritmus *szétválasztás* eljárása.

5.2. Az algoritmus korlátozási lépése

A *korlátozás* eljárásban (17. ábra) először egy megvalósíthatósági vizsgálatot (*feasibility test*) végzünk el. Ha a részfeladat megvalósítható (*feasible*), akkor meghatározunk egy alsó korlátot a részfeladatból elérhető megoldások működési idejére. Ha a köztes termékek várakozási ideje két taszk között nincs korlátozva, akkor egy ütemezési-gráf által reprezentált megoldás akkor és csak akkor megvalósítható, ha a gráf nem tartalmaz irányított kört. Tehát a megvalósíthatósági vizsgálat elvégezhető egy körkereső algoritmus segítségével. Érdemes megjegyezni, hogy létezik polinomiális idejű körkereső algoritmus (a Mellékletben található egy részletes körkereső algoritmus). Ha a köztes termékek várakozási ideje korlátozva van (ZW tárolási stratégia), akkor további vizsgálat szükséges, például egy lineáris programozási (*linear programming*, LP) feladat megoldása.

```
procedure korlátozás (G(N, A), bound)
begin
    cycle_search (G(N, A));
    if no cycle then
        bound = longest_path (G(N, A));
    else
        bound = ∞;
end
```

17. ábra: Az ütemezési algoritmus *korlátozás* eljárása.

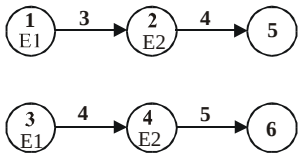
Egy részfeladat S-gráfja mindig részgráfja bármely leszármazott részfeladat S-gráfjának. Mivel egy új (nem negatív súlyú) él hozzáadása egy gráfhoz nem csökkentheti a leghosszabb utat, ezért egy ütemezési-gráf bármely részgráfjához tartozó leghosszabb út alsó korlát az ütemezési-gráf leghosszabb útjára. Továbbá egy ütemezési-gráfhoz tartozó leghosszabb út szintén egy alsó korlátja a hozzá tartozó megoldás értékére, mivel az (i, j) él egy S-gráfban azt jelenti, hogy a j csomópont által reprezentált taszk elvégzése leghamarabb $c(i, j)$ idővel később kezdődhet el az i csomópont által reprezentált taszk elkezdéséhez képest. Ebből következik, hogy egy ütemezési-gráf bármely részgráfjához tartozó leghosszabb út alsó korlátot ad az ütemezési-gráfhoz tartozó megoldás értékére. (A Mellékletben található egy részletes leghosszabb út kereső algoritmus.)

A szétválasztás és korlátozás algoritmus sok részfeladatánál, főleg a kereső fa gyökérhez közeli szintjein, a leghosszabb út nem ad éles alsó korlátot egyszerűen azért, mert a gráf nem eléggé összekötött, mivel nincs vagy csak kevés ütemezési-él szerepel a gráfban. Ezt az alsó korlátot általában élesíteni tudjuk egy speciális, a leghosszabb út kereső algoritmus eredményeit felhasználó LP feladat megoldásával (7.2. fejezet), a korlátozás eljárás számolási idejének növekedése fejében.

Ha a köztes anyagok két taszk közötti várakozási ideje korlátozott, mindenképpen LP feladatot kell megoldanunk a megvalósíthatósági vizsgálathoz, illetve az alsó korlát meghatározásához. A szétválasztás és korlátozás szabályai alapján, ha egy részfeladat alsó korlátja nagyobb, mint az aktuális felső korlát (például az aktuálisan megtalált legjobb megoldás) vagy az S-gráf irányított kört tartalmaz, akkor a részfeladatot és a keresőfa hozzátartozó csomópontját eldobhatjuk.

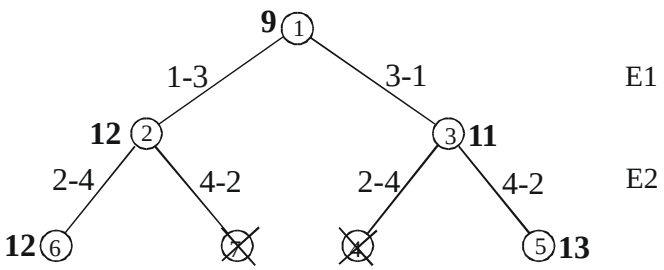
5.3. Az algoritmus működésének szemléltetése

Sanmartí és társai (2002) által publikált alapalgoritmus a leghosszabb út kereső algoritmust használja, ezért az algoritmus működésének bemutatásához mi is ezt használjuk. A szemléltető példa recept-gráfja a 18. ábrán látható, ahol két terméket kell előállítani két lépésben két berendezéssel.

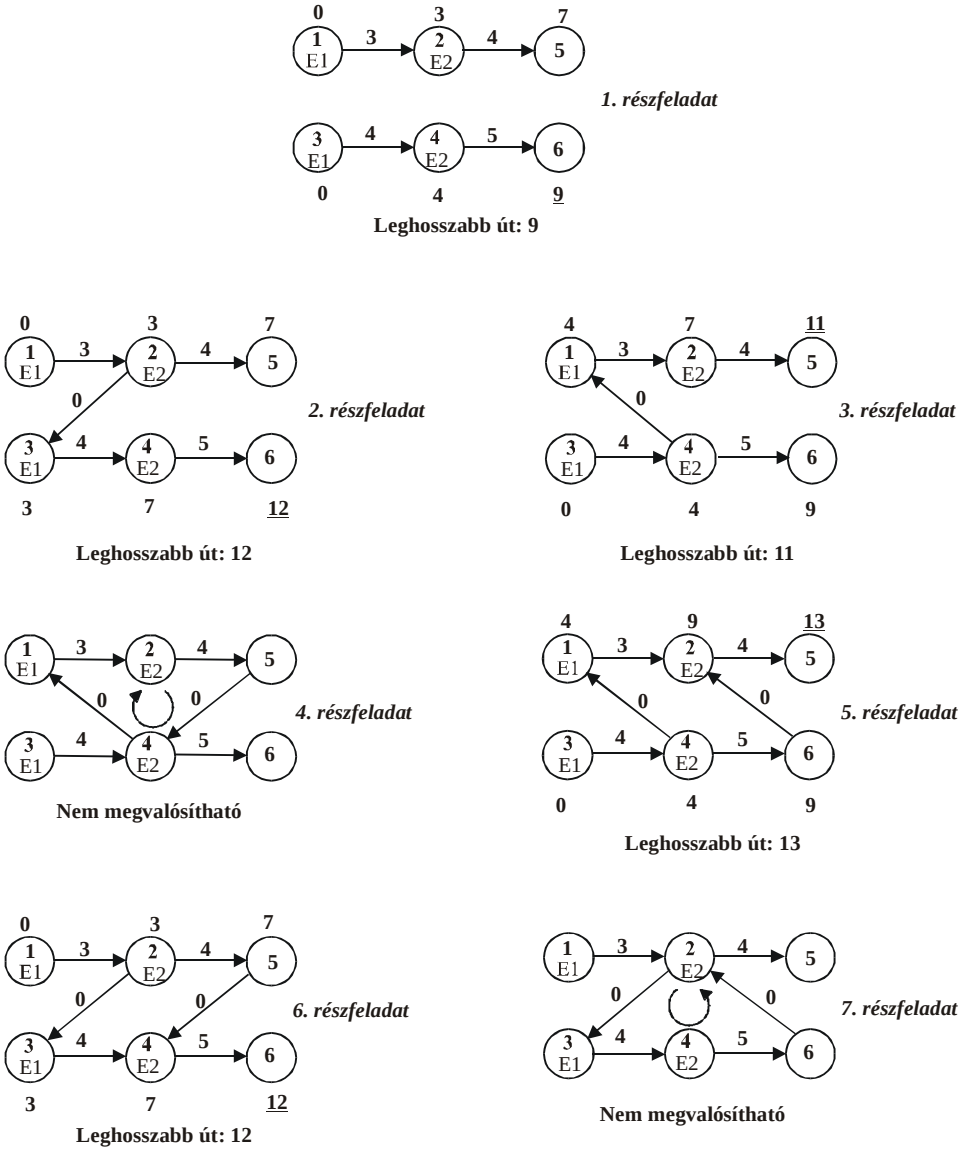


18. ábra: Recept-gráf a szemléltető példához.

A példához tartozó kereső fa a 19. ábrán látható. A csomópontok a részfeladatokat jelölik és a generálási sorrend szerint számoztuk őket. A szétválasztásokat (döntéseket) a taszkok sorrendjével jelöljük, amelyeket meghatároznak. Az alsó korlát minden részfeladat csomópontja mellett vastagított számmal szerepel. A nem megvalósítható részfeladatok csomópontjai át vannak húzva (pl. 4. részfeladat). A részfeladatokhoz tartozó S-gráfok a 20. ábrán láthatóak.



19. ábra: A szemléltető példához tartozó keresőfa.



20. ábra: A szemléltető példa részfeladataihoz tartozó S-gráfok.

A keresőfa gyökeréhez (1. részfeladat) tartozik a recept-gráf, itt a leghosszabb út hossza, azaz az alsó korlát értéke 9. Ebből a részfeladatból két új részfeladatot származtatunk (2. és 3. részfeladat) az E1 berendezés ütemezésével: 1-3 illetve 3-1 sorrendekkel, 12 illetve 11 alsó korláttal. A legjobb alsó korláttal a 3. részfeladat rendelkezik, tehát ennek szétválasztásával folytatjuk az algoritmust. Ebből a részfeladatból a szétválasztás után két újabb részfeladat keletkezik (4. és 5. részfeladat) az E2 berendezés 2-4 illetve 4-2 sorrend választásával. Mind a két részfeladat egy teljesen beütemezett megoldást jelöl, azaz egy új felső korlátot kaphatunk a feladatunkhoz. A 4. részfeladat S-gráfja kört tartalmaz, tehát nem megoldható ezért eldobjuk. Az 5. részfeladat S-gráfja ütemezési-gráf, ahol a

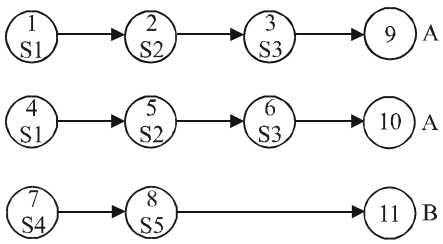
leghosszabb út 13. Ez jelenti az új felső korlátunkat az optimumhoz. Az algoritmus a 2. részfeladat szétválasztásával folytatódik, ahol az E2 berendezésről dönthetünk 2-4 illetve 4-2 ütemezési sorrendekkel. Így két részfeladat generálódik (6. és 7. részfeladat), amelyek teljesen beütemezett megoldást jelölnek. A 6. részfeladat egy megoldást jelöl, amelynek értéke 12, ami jobb, mint az aktuális felső korlátunk, ezért ez lesz az új felső korlátunk. A 7. részfeladat kiértékelése nem szükséges, mert a szülőjének (2. részfeladat) alsó korlátja nem jobb, mint az aktuális felső korlát. Ha mégis kiértékeljük, akkor kiderül, hogy a hozzá tartozó S-gráf irányított kört tartalmaz, tehát nem megvalósítható megoldást jelöl.

6. Az alapalgoritmus kiterjesztése: egy termékből több batch előállítása

Azok az ütemezési feladatok, amelyekben a termékekből több mint egy batch-et kell előállítani egyszerűen kezelhetőek, minden batch egy külön terméknek való tekintésével. Bár az előző fejezetben bemutatott algoritmus képes ennek a modellnek a megoldására, ez nem elég hatékony, ha a batch-ek száma nagyon nagy. Az alapalgoritmusba beépíthető eszközök segítségével viszont elérhetünk olyan mértékű gyorsítást, amely lehetővé teszi nagy méretű problémák megoldását is. Először egy egyszerű példán keresztül mutatjuk be, hogy miért nem hatékony a batch-ek külön termékként való kezelése.

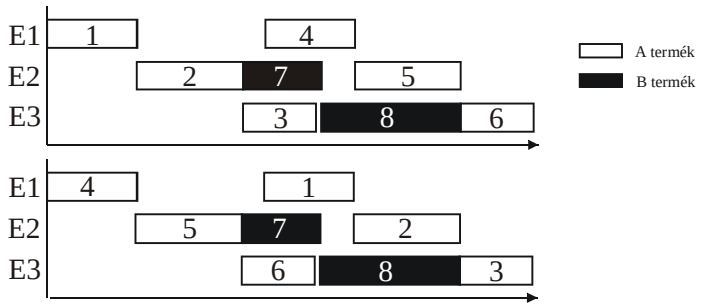
4. példa

Tegyük fel, hogy két batch-nyi A terméket és egy batch-nyi B-t kell előállítani. Az A terméket három egymást követő lépésben lehet előállítani sorrendben az S1, S2 és S3 halmazban lévő berendezések valamelyikének segítségével. A B terméket két egymást követő lépésben lehet előállítani sorrendben az S4 és az S5 halmazokban lévő berendezések valamelyikével. A példa recept-gráfja a 21. ábrán látható.



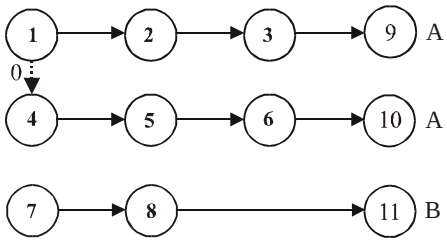
21. ábra: A 4. példa recept-gráfja.

Tegyük fel, hogy $E1 \in S1$, $E2 \in S2 \cap S4$ és $E3 \in S5$. Ebben az esetben létezik két technikailag azonos ütemezés, amelyek a 22. ábrán láthatóak. Ezek közül csak az egyiket kell figyelembe vennünk annak ellenére, hogy az alapalgoritmus mind a kettőt generálhatja.



22. ábra: Két technikailag azonos ütemezés.

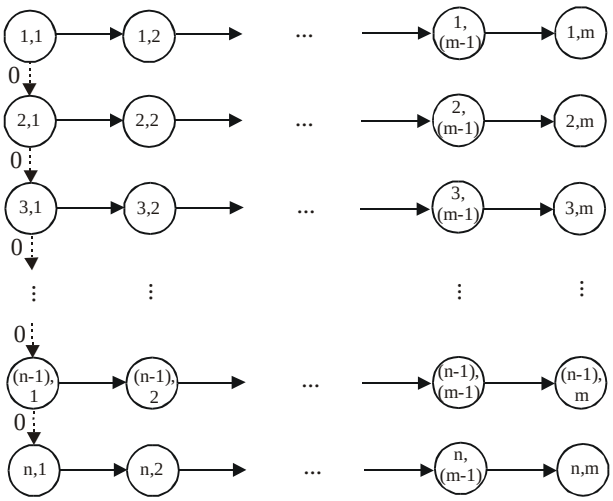
Ahogy a példa is mutatja a módszer nem hatékony, mert létezhetnek olyan ekvivalens ütemezések, amelyek csak a termékekhez tartozó batch-ek sorrendjében különböznek. Ezt a redundanciát megszüntethetjük új feltételek bevezetésével. Ha például feltesszük, hogy a 4. tevékenység nem kezdődhet hamarabb, mint az 1. tevékenység, akkor a 22. ábrán látható redundancia kizárható. Ezt a feltételt egy nulla súlyú él behúzásával az 1. és a 4. csomópontok közé egyszerűen megadhatjuk (23. ábra). Meg kell jegyezni, hogy ez az él nem tartozik a recept-gráfhoz vagy az ütemezési-gráfhoz, továbbá az ilyen típusú éleket segéd-éleknek nevezzük és szaggatott vonallal jelöljük.



23. ábra: A 4. példa recept-gráfja segéd-éllel kiegészítve, amely az A termék batch-einek sorrendjét mutatja.

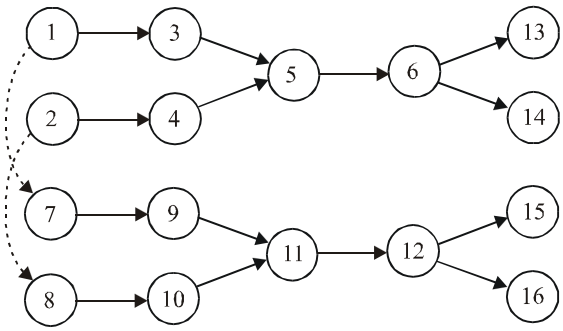
Ha több mint két batch-nyit kell gyártani egy termékből, akkor a redundancia sokkal nagyobb. Ha például *n* batch-nyit kell egy termékből előállítani, akkor *n* faktoriális számú technikailag azonos megoldás lehet csak ezen termék batch-jeinek sorrendjét tekintve. A batch-ekhez tartozó első tevékenységek kezdetének sorrendbe állításával (például a második batch nem kezdődhet hamarabb az elsőnél) a batch-ek szükségtelen permutációját ki tudjuk zárni. A sorrend jelölését az S-gráfban legegyszerűbben nulla súlyú élekkel oldhatjuk meg, ahogy az a 24. ábrán látható. Természetesen ezek a feltételek nem

zárják ki az optimális megoldást a vizsgálatból, mivel minden optimális megoldáshoz létezik *n*!-1 számú olyan különböző optimális megoldás, amelyek mindegyike csak az adott termékhez tartozó batch-ek sorrendjében különbözik egymástól.



24. ábra: Recept-gráf segéd-élekkel kiegészítve, ha n batch-nyit kell egy termékből gyártani.

Összetett recept esetén (a recept tartalmaz több kimenettel, illetve bemenettel rendelkező taszkot), ha egy batch-nek több első tevékenysége is van, akkor a redundanciát hasonlóképpen segéd-élekkel oldhatjuk fel, összekötve a különböző batch-ekhez tartozó megfelelő első csomópontokat (25. ábra).



25. ábra: Recept-gráf segéd-élekkel kiegészítve összetett receptnél két batch esetén.

Ahhoz, hogy ezt a kiterjesztést megtehessük bizonyítanunk kell, hogy bármely megoldáshoz létezik az ütemezési-gráfnak segéd-élekkel való olyan kiterjesztése, mely nem változtatja meg a megoldás értékét és a megoldás megvalósítható marad.

1. Állítás: Egy tetszőleges ütemezési-gráfot ki lehet egészíteni az egy termékhez tartozó batch-ek első csomópontjai között segéd-élekkel, nem okoznak kört és nem növelik meg a leghosszabb út hosszát.
Bizonyítás: Lásd a Mellékletben.

6.1. Batch kezelés különböző esetekben

Az előbbiekben leírt módszer általános és bármikor alkalmazható, de léteznek olyan esetek is, amikor a redundanciát kizáró feltételeket tovább tudjuk élesíteni, illetve az algoritmus hatékonysága érdekében további feltételeket lehet bevezetni. Ilyen eset például, amikor minden taszkhoz pontosan egy berendezés áll rendelkezésre.

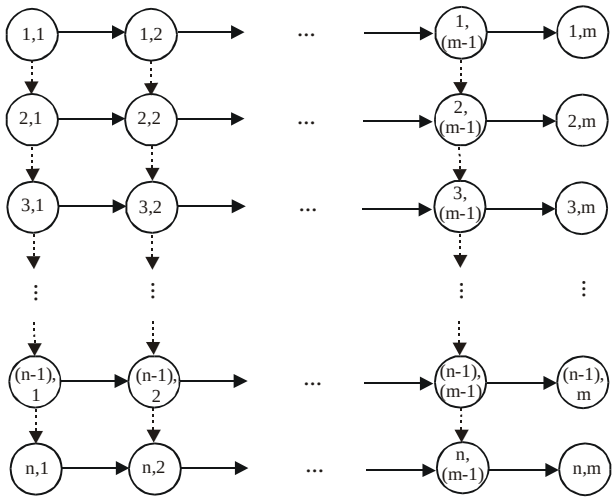
6.1.1. Minden taszkhoz pontosan egy berendezés tartozik

Tegyük fel, hogy minden taszk elvégzéséhez pontosan egy berendezés áll rendelkezésre. Ebben az esetben igaz a következő állítás.

2. Állítás: Ha minden taszkhoz pontosan egy berendezés tartozik, a segéd-éleket a batch-ekhez tartozó összes ekvivalens csomópont között fel lehet venni, az elsővel azonos irányban.
Bizonyítás: Lásd a Mellékletben.

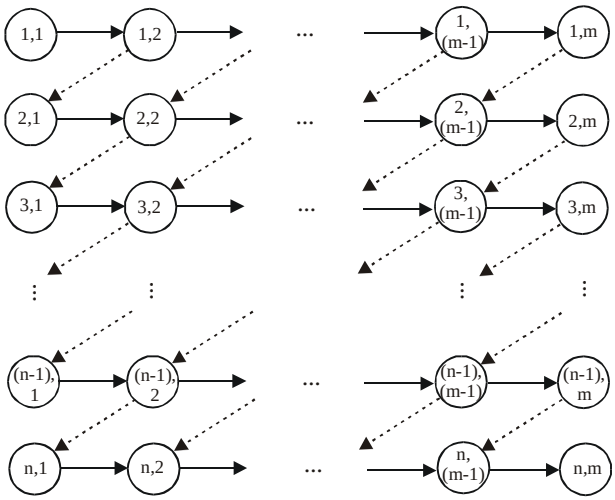
A 2. Állításból következik, hogy minden megvalósítható megoldásban a második tevékenységek sorrendje ugyanolyan lesz, mint az első tevékenységeké, a harmadik tevékenységek sorrendje ugyanolyan lesz, mint a második tevékenységeké, és így tovább. Tehát a batch-ekhez tartozó összes tevékenységet ugyanúgy rendezhetjük. Egy ilyen módon rendezett segéd-élekkel kiegészített recept-gráf a 26. ábrán látható.

NIS tárolási stratégia esetében (amit az S-gráf alapértelmezésben feltételez) a redundanciát kizáró feltételeket tovább lehet élesíteni.



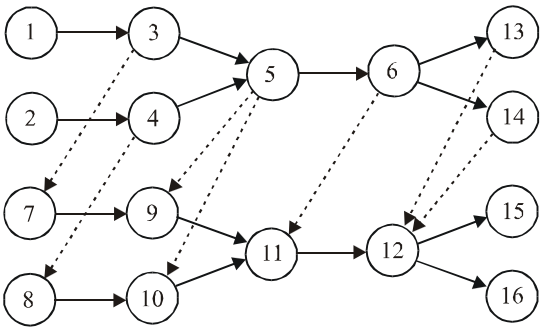
26. ábra: Recept-gráf segéd-élekkel kiegészítve, ha egy termék minden taszkját egy berendezés hajthatja végre.

3. Állítás: Ha minden taszkhoz pontosan egy berendezés tartozik, a segéd-él kezdőpontja a recept szerinti következő tevékenységet jelölő csomópontja lehet.
Bizonyítás: Lásd a Mellékletben.



27. ábra: Recept-gráf módosított segéd-élekkel, ha egy termék minden taszkját egy berendezés hajthatja végre.

Tegyük fel, hogy az (i, j) csomópont az i-edik batch j-edik tevékenységét jelöli. Ekkor segéd-él kezdőpontja a recept szerinti következő tevékenységet jelölő $(i, j+1)$ csomópontja is lehet, ahogy ez a 27. ábrán látható. Az ilyen módon transzformált segéd-élek nem csak a redundanciát csökkentik, hanem élesebbé teszik az alsó korlátot és így további gyorsítást eredményeznek.



28. ábra: Összetett recept recept-gráfja módosított segéd-élekkel, ha egy termék minden taszkját egy berendezés hajthatja végre.

6.1.2. Általános eset

A gyakorlatban általában a 6.1.1. fejezetben leírt tulajdonságú taszkok mellett léteznek olyan taszkok is a receptben, amelyek különböző berendezésekkel hajthatók végre. Az egy termékhez tartozó első tevékenységek csomópontjait ezekben az esetekben is összeláncolhatjuk, ahogy az a 24. és a 25. ábrákon bemutattuk, valamint azon tevékenységek csomópontjait is összeláncolhatjuk, amelyek olyan taszkokhoz tartoznak, amik teljesítik a következő két tulajdonságot.

(P1) A taszkhoz pontosan egy berendezés tartozik.

(P2) A receptben az összes megelőző taszk teljesíti a P1 tulajdonságot.

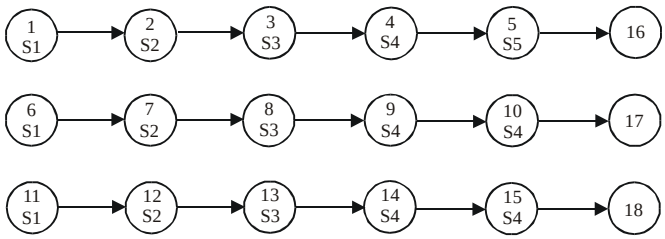
Abban az esetben, ha van olyan taszk, mely teljesíti a tulajdonságokat, akkor a segéd-éleket transzformálhatjuk a 6.1.1. fejezetben leírt módon.

4. Állítás: Ha egy taszk teljesíti a P1 és P2 tulajdonságot, akkor a hozzá tartozó tevékenységek csomópontjai transzformált segéd-élekkel összeláncolhatók a receptben őt megelőzővel azonos irányban.

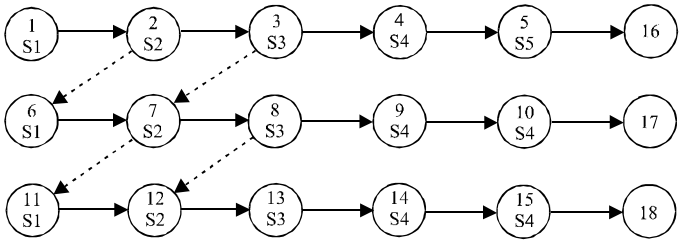
Bizonyítás: Lásd a Mellékletben.

5. példa

A 29. ábrán látható recept-gráf esetén feltételezzük, hogy S1={E1}, S2={E3}, S3={E4, E5}, S4={E1} és S5={E2}. Ekkor az első, a második. a negyedik és az ötödik taszk teljesíti a P1 tulajdonságot, továbbá az első, a második és a harmadik taszk teljesíti a P2 tulajdonságot. Tehát az első és a második taszk taszk-csomópontjait összeláncolhatjuk módosított segéd-élekkel (lásd 30. ábra).



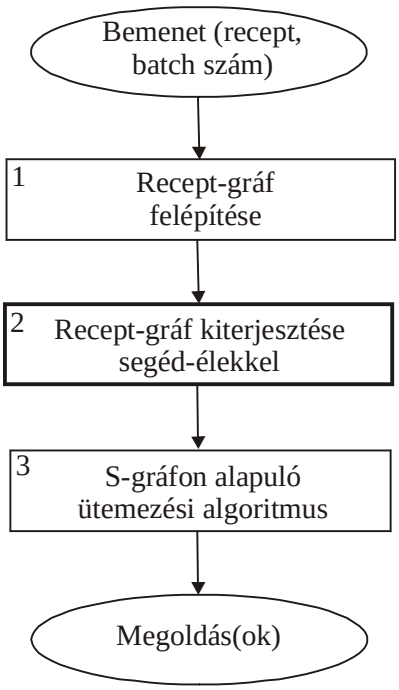
29. ábra: Az 5. példa recept-gráfja.



30. ábra: Az 5. példa recept-gráfja segéd élekkel kiegészítve.

6.2. A batch kezelés beépítése az algoritmusba

Az alapalgoritmus (5. fejezet) két fő lépésből áll: a recept-gráf felépítése valamint az S-gráfon alapuló szétválasztás és korlátozás típusú ütemező algoritmus (első és harmadik lépés a 31. ábrán). Batch kezelés esetén ez az algoritmus egy köztes lépéssel, a segéd-élek behúzásával (második lépés a 31. ábrán), bővül. Bár a bemutatott gyorsítási módszer az S-gráfon alapuló alapalgoritmushoz készült, valójában nem S-gráf specifikus és lehet használni más ütemezési algoritmusokhoz is.



31. ábra: A megoldás menete segéd-élek használata esetén.

7. Gyorsítási módszerek az alapalgoritmushoz

Az előzőekben bemutatott S-gráf leírás és a hozzá tartozó korlátozás és szétválasztás algoritmus hatékonyságát növelni tudjuk az S-gráf és az alapalgoritmus nagy szabadsági fokának köszönhetően. Az algoritmus szétválasztás lépésében nem csak az 5. fejezetben említett a berendezések leterheltsége szerinti sorrendben való ütemezéssel tudjuk a megoldás menetét gyorsítani, hanem például a nem megvalósítható részfeladatokat (kört tartalmaz a részfeladat S-gráfja) már a kialakulásuk előtt fel lehet ismerni egy egyszerű algoritmus segítségével (7.1. fejezet). Ezen kívül a korlátozás lépésében nem mindig a leghosszabb út kereső algoritmus használata bizonyult a leghatékonyabb módszernek, az általa adott alsó korlátot gyakran tudjuk pontosítani (7.2. fejezet). Ez természetesen a korlátozás sebességének csökkenését vonja maga után, viszont a teljes algoritmusra nézve gyorsítást is jelenthet.

7.1. Előzetes körfelismerés

Az (i, j) él egy S-gráfban azt jelenti, hogy a j csomópont által reprezentált taszk elvégzése leghamarabb $c(i, j)$ idővel később kezdődhet el az i csomópont által reprezentált taszk elkezdéséhez képest, ahol $c(i, j) \geq 0$. Ha egy részfeladat S-gráfja tartalmaz egy olyan irányított kört, amelyik nem tartalmaz segéd-élet, akkor ez a kör holtpontot (*deadlock*) jelöl, azaz az aktuális részfeladat nem megvalósítható. Ha az S-gráf minden irányított köre tartalmaz legalább egy segéd-élet, akkor a részfeladat még nem feltétlenül nem megvalósítható, de megsérti a 6 fejezetben leírt batch kezelési szabályainkat, tehát a részfeladatot eldobhatjuk, nincs szükség a további vizsgálatára.

Egy irányított gráfban, esetünkben egy S-gráfban, az irányított köröket egy egyszerű algoritmussal lehet detektálni, amely hatékony módszer a nem megvalósítható részfeladatok felismeréséhez. Sok esetben, bár az S-gráf nem tartalmaz kört (azaz megvalósítható), nagyon gyakori, hogy a keresőfában lefelé indulva előbb utóbb kört tartalmazó S-gráfhoz jutunk, azaz ebből a részfeladatból nem érhetünk el ütemezési-gráfhoz. Ebből következik, hogy ennek a

tulajdonságnak a felismerése nagy segítséget nyújthat a keresésben, gyorsíthatja azt. A célunk egy olyan előretekintő stratégia kidolgozása volt, amellyel fel lehet ismerni, hogy egy részfeladat S-gráfjából származtatható S-gráfokban biztosan keletkezik-e kör a további keresés folyamán. Ezen kívül nagy segítséget jelent, ha felismerjük, hogy melyek azok a döntések, melyeket egy leszármazott részfeladatban sem szabad meghoznunk, mert az kört eredményez. Ezzel az úgynevezett előzetes körfelismerő módszerrel a keresési teret nagy mértékben csökkenteni tudjuk, azaz a megvizsgálandó részfeladatok számát is csökkentjük. Az előzetes körfelismerő algoritmusunk a következő állításon alapul.

5. Állítás: Tegyük fel, hogy a *k* berendezés hozzá van rendelve az *i* taszkhoz az aktuális részfeladatban. Ha az *i* taszk elérhető az aktuális részfeladathoz tartozó részben beütemezett S-gráfban egy irányított úton egy *j* taszkból, akkor nincs olyan megvalósítható megoldás, amelyik az aktuális részfeladaton alapul és ahol a *j* taszk követi az *i* taszkot a *k* berendezés ütemezési sorrendjében.
Bizonyítás: Lásd a Mellékletben.

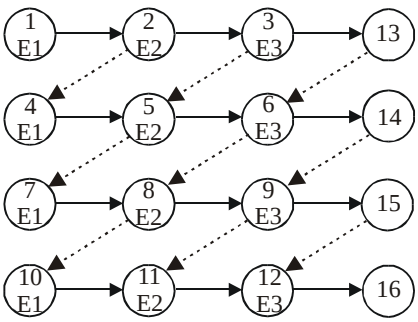
Ezt az állítást fel tudjuk használni az alapalgoritmus kiegészítéséhez a következő módon. Tegyük fel, hogy csak a *k* berendezés áll rendelkezésre a *j* még nem ütemezett csomóponthoz (taszkhoz) és ebből a csomópontból elérhető egy olyan ütemezett csomópont, amelyhez a *k* berendezést már hozzárendeltük (a Mellékletben található részletes útkereső algoritmus). Ebben az esetben biztos, hogy nem érhetünk el ütemezési-gráfot az aktuális S-gráfból, mivel az alapalgoritmus a taszkot mindig a berendezés aktuálisan legutolsó lépéseként ütemezi. Ha viszont több berendezés is rendelkezésre áll a *j* még nem ütemezett csomóponthoz, akkor a kör keletkezését megelőzhetjük egy másik berendezés hozzárendelésével. Ebben az esetben a *k* berendezést a részfeladat egyetlen leszármazott részfeladatában sem rendelhetjük hozzá a *j* csomóponthoz, azaz $N_k = N_k \setminus j$. Az algoritmusban elegendő azt ellenőrizni, hogy az aktuálisan ütemezett csomópont elérhető-e valamely olyan még nem ütemezett csomópontból, melyhez az aktuálisan ütemezett berendezéssel hozzá lehet rendelni. Az algoritmus leírása a 32. ábrán látható.

```
procedure cycle_prediction(PP, k)
jelölések: n: berendezések száma
           N_i (i = 1,2,...,n): az i berendezéshez rendelhető csomópontok halmaza
           SOUN: nem ütemezett csomópontok halmaza
           last_node: (i, j) rendezett pár, ahol i egy berendezés és j egy csomópont
           G(N, A_1, A_2): részben ütemezett S-gráf
           bound: alsó korlát
           PP = (G(N, A_1, A_2), bound, last_node, SOUN, N_1, N_2, ...,N_n)
           k: aktuálisan ütemezett csomópont
begin
    SO = N_k ∩ SOUN;
    for all j ∈ SO do
        path_search(G(N, A_1∪A_2), k, j);
        if path then
            if j ∈ N_m, ahol m = 1,2,...,n és m ≠ k then
                N_k = N_k \ j;
            else
                return cycle;
    return no_cycle;
end
```

32. ábra: Előzetes körfelismerő algoritmus.

6. példa

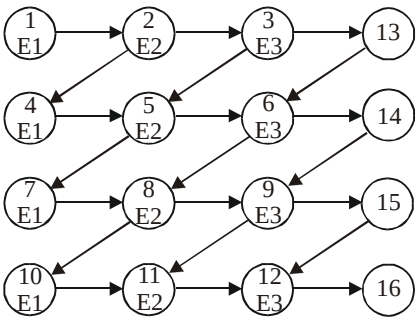
Tegyük fel, hogy négy bacth-nyit kell előállítani egy termékből három egymást követő lépésben, ahol az első taszkot az E1, a második taszkot az E2 és a harmadikat az E3 berendezéssel lehet csak végrehajtani. A példához tartozó segéd-élekkel kiegészített recept-gráf a 33. ábrán látható.



33. ábra: Az 6. példa recept-gráfja segéd-élekkel kiegészítve.

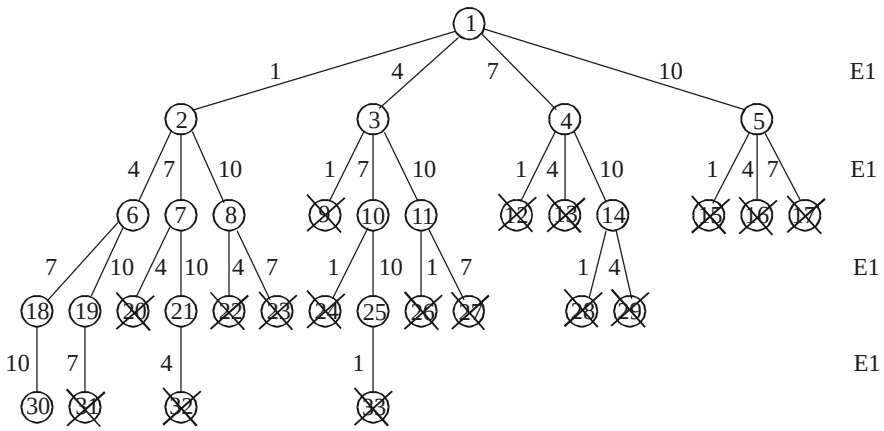
Tekintsünk azt a részfeladatot, ahol az E1 berendezés először a 4. tevékenységet hajtja végre. Ebből a részfeladatból nem érhető el megvalósítható megoldás mivel az 1. tevékenység csak az E1 berendezéssel lehet végrehajtani és a döntésünk szerint csak a 4. tevékenység befejezése után. Ez ellentmond az előző fejezetben bevezetett batch kezelésnek.

Egyszerűen belátható, hogy a 6. példához egyetlen megvalósítható megoldás létezik, amikor minden berendezés először az első batch-en dolgozik, utána a másodikon, majd a harmadikon, végül a negyediken. A megoldás ütemezési-gráfja a 34. ábrán látható.

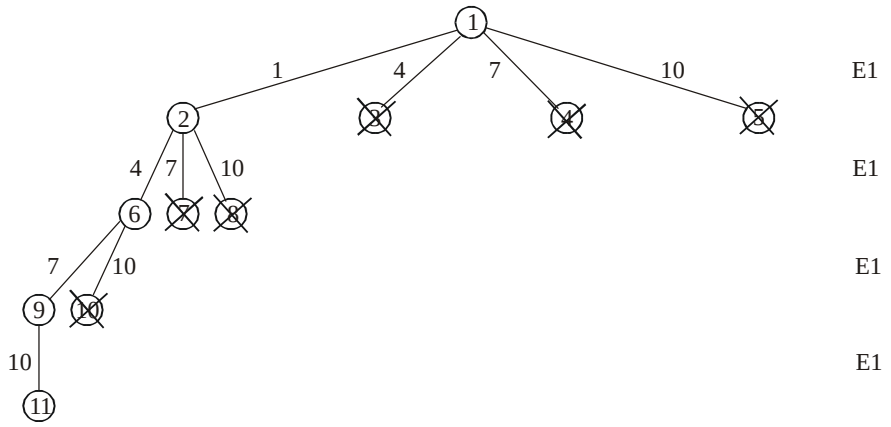


34. ábra: A 6. példa ütemezési-gráfja.

Tegyük fel, hogy az algoritmus futása során nem tudunk egy részfeladatot sem eldobni az alsó korlát alapján, azaz tételezzük fel a legrosszabb esetet. Ekkor a 35. ábra mutatja az E1 berendezés ütemezésének menetét az előzetes körfelismerő algoritmus használata nélkül, valamint a 36. ábra mutatja a keresőfa ugyanezen részét az előzetes körfelismerő algoritmus használata esetén. Az első esetben 33 részfeladatot kell generálnunk és az alsó korlátot 15-ször kell meghatároznunk. A második esetben 11 részfeladat van és 5-ször kell a korlátozás függvényt meghívni. Ez azt jelenti, hogy a példában a gyorsítás használatával a keresési teret a harmadára csökkentettük az E1 berendezés ütemezésénél.



35. ábra: A 6. példa keresőfájának részlete: az E1 berendezés ütemezése az előzetes körfelismerő algoritmus használata nélkül (az áthúzott csúcsok nem megvalósíthatók).



36. ábra: A 6. példa keresőfájának részlete: az E1 berendezés ütemezése előzetes körfelismerő algoritmus használatával (az áthúzott csúcsok nem megvalósíthatók).

Példánkban a kereső fa gyökeréhez (1. részfeladat) tartozik a recept-gráf és ebből a részfeladatból négy új részfeladat keletkezik (2., 3., 4. és 5. részfeladat). Az előzetes körfelismerő algoritmus használata nélkül egyik részfeladatot sem dobhatjuk el (35. ábra), mivel egyik részfeladat S-gráfja sem tartalmaz kört. Az algoritmust használatával a négy részfeladatból hármat (3., 4. és 5. részfeladat) eldobhatunk (36. ábra), mivel ezekben a részfeladatokban az 1. tevékenység nincs ütemezve. Ezt a tevékenységet csak az E1 berendezéssel lehet végrehajtani, viszont a döntésünk szerint a berendezésnek előbb kell végrehajtania egy a batch kezelés miatt csak később következő tevékenységet (4., 7. illetve 11. tevékenység), tehát a részfeladataink ellentmondásosak, további vizsgálatuk nem szükséges. Az előzetes körfelismerő algoritmus használata nélkül a teljes keresőfában 97 darab részfeladat van, ami az előzetes körfelismerő algoritmus használatával 31-re csökken. Mind a két esetben az ütemezési algoritmus pontosan egy megvalósítható megoldást talál és ezen megoldások azonosak.

7.2. LP modell a korlátozás élesítéséhez

A leghosszabb út kereső algoritmus nagyobb méretű feladatoknál, ha az S-gráfban kevés az él, azaz amíg közel vagyunk a keresőfa gyökeréhez, nem ad megfelelően éles alsó korlátot. Az aktuális részfeladat S-gráfjából elérhető ütemezési-gráfokhoz tartozó leghosszabb út hosszát jól lehet alulról becsülni egy LP modell segítségével.

Az LP modell felírásához be kell vezetnünk a következő jelöléseket. Tegyük fel, hogy bármely részfeladat S-gráfjának j csomópontjába vezető leghosszabb út (L_j) adott minden $j \in N$-re. Ezen kívül jelölje $\overline{M}_i$ ($i = 1,2,...,n$) a csomópontoknak azon halmazát, amely az alábbi formalizmussal adott

$$\overline{M}_i = \{ j : j \in N_i \setminus (\bigcup_{k=1}^{n} M_k) \setminus (\bigcup_{\substack{k=1 \\ k \neq i}}^{n} N_k) \},$$

azaz azon csomópontokat, amelyekhez még nem történt berendezés hozzárendelés és csak az *i* berendezéssel hajthatók végre.

Minden részfeladatra meg tudjuk határozni azt a c_i ($i = 1,2,...,n$) időpontot, amely időpontnál hamarabb az *i* berendezés nem fejezi be a munkáját bármely, az aktuális részfeladathoz tartozó megoldás esetén. Ez az időpont akkor a legkisebb, ha az *i* berendezés csak azokhoz a csomópontokhoz lesz a későbbiekhez hozzárendelve, amelyekhez más berendezés nem használható fel, azaz ez az időpont függ a már beütemezett taszkoktól (már történt berendezés hozzárendelés) és azon még be nem ütemezettektől (még nem történt berendezés hozzárendelés), amelyek végrehajtására csak az *i* berendezés használható fel. A c_i értékek a következőképpen határozzuk meg. Először meghatározzuk, hogy az *i* berendezés leghamarabb mikor tudja befejezni a munkáját a már beütemezett taszkokon ($\max_{j \in M_i}(L_j + t_{ij})$, ahol t_{ij} a j taszk működési ideje az *i* berendezéssel használata esetén) és ezt összehasonlítjuk a még be nem ütemezett taszkok közül annak az idejével, ahol az *i* berendezés leghamarabb elkezdhet dolgozni ($\min_{j \in \overline{M}_i}(L_j)$). Ezek után a két érték közül a nagyobbhoz hozzáadjuk azon még be nem ütemezett taszkok működési idejét, amelyet csak az *i* berendezéssel lehet végrehajtani ($\sum_{j \in \overline{M}_i} t_{ij}$). Így c_i ($i = 1,2,...,n$) meg tudjuk határozni a következő képlettel:

$$c_i = \max(\max_{j \in M_i}(L_j + t_{ij}), \min_{j \in \overline{M}_i}(L_j)) + \sum_{j \in M_i} t_{ij}$$

A c_i ($i = 1,2,...,n$) értékek meghatározásával az S-gráf csomópontjainak egy részét megvizsgáltuk. Ezek után már csak azok a be nem ütemezett

csomópontok ($\overline{N} = (\bigcup_{i=1}^{n} N_i) \setminus (\bigcup_{i=1}^{n} M_i) \setminus (\bigcup_{i=1}^{n} \overline{M}_i)$) maradtak hátra, amelyek még nincsenek beütemezve és végrehajtásukhoz több berendezés is rendelkezésre áll. Minden ilyen taszkhoz bevezetünk *n* darab (berendezések száma) pozitív értékű, valós típusú változót, (x_{ij}, $i = 1,2,...,n$, $j \in \overline{N}$), amelyek azt jelölik, hogy optimális esetben melyik berendezés mennyi időt dolgozik a taszkon, ha megengedjük, hogy egy taszkot több berendezés is végrehajthasson, akár időben eltolva egymástól. Ezen változók és a c_i paraméterek segítségével fel tudjuk írni az egyes berendezések működésének befejezési idejét ($c_i + \sum_{j \in \overline{N}} x_{ij}$, $i = 1,2,...,n$), amely természetesen minden berendezésre kisebb vagy egyenlő, mint a teljes rendszer működési ideje. Az x_{ij} változók értéke olyan határok közt változhat, hogy az egy taszkhoz tartozó berendezések együttes működése legalább a *j* taszk működéséhez szükséges teljesítményt kiadja ($\sum_{i \in S_j} x_{ij} / t_{ij} \geq 1$, ahol $j \in \overline{N}$). Így az eredeti feladatunkhoz képest nem vesszük figyelembe, a receptből és a berendezések további ütemezésének működési sorrendjéből adódó késleltetéseket (például a berendezések tisztítási idejét) és megvalósíthatóságot, valamint azt a megkötést, hogy a megoldásban egy taszkot csak egy berendezés hajthat végre.

Az előzőekből egyértelműen következik, hogy alábbi LP feladat megoldása alsó korlátot ad az aktuális részfeladathoz.

$$\min X$$

feltéve, hogy

$$c_i + \sum_{j \in \overline{N}} x_{ij} \leq X \qquad (i = 1,2,...,n),$$

$$\sum_{i \in S_j} \frac{x_{ij}}{t_{ij}} \geq 1 \qquad (j \in \overline{N}) \text{ és}$$

$$x_{ij} \geq 0 \qquad (i = 1,2,...,n,\ j \in \overline{N}),$$

ahol

n a berendezések száma,

N_t taszk-csomópontok halmaza,

$\overline{N}$ a még be nem ütemezett, több berendezéssel végrehajtható csomópontok halmaza,

S_j a j taszkot végrehajtható berendezések halmaza ($j = 1,2,...,|N_t|$),

x_{ij} (változó) az i berendezés működési ideje a j taszkban ($i = 1,2,...,n,\ j = \overline{N}$)

t_{ij} a j taszk működési ideje az i berendezéssel használata esetén ($i = 1,2,...,n,\ j = 1,2,...,|N_t|$) és

X (változó) alsó korlát a részfeladatra.

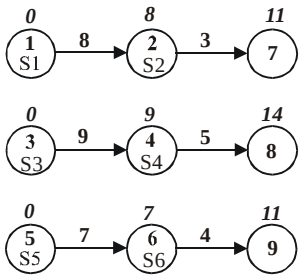
7. példa

Három terméket (A, B, C) kell előállítani három berendezéssel (E1, E2, E3), minden terméből egy batch-nyit. A receptek a 3. táblázatban adottak.

3. táblázat: A 7. példa receptje.

	A		B		C	
Taszk	Beren-dezés	Idő (óra)	Beren-dezés.	Idő (óra)	Beren-dezés	Idő (óra)
1	E1	8	E1	9	E1	7
			E2	11	E2	7
2	E2	15	E3	5	E3	4
	E3	5				

A példa recept-gráfja a 37. ábrán adott, ahol a csomópontokba vezető leghosszabb út értéke szerepel minden csomópont felett dőlt számokkal látható.



37. ábra: A 7. példa recept-gráfja.

Az LP modell a kereső fa gyökerére (azaz a recept-gráfra) a következőképpen határozható meg. Mivel $\overline{M}_1 = \{1\}$, $\overline{M}_2 = \varnothing$, $\overline{M}_3 = \{4,6\}$, $\overline{N} = \{2,3,5\}$, $c_1 = 8$, $c_2 = 0$, és $c_3 = 16$, az alábbi LP modellt kell megoldanunk:

$$\min X$$

feltéve, hogy

$$8 + x_{13} + x_{15} \le X$$

$$x_{22} + x_{23} + x_{25} \le X$$

$$16 + x_{32} \le X$$

$$\frac{x_{13}}{9} + \frac{x_{23}}{11} \ge 1$$

$$\frac{x_{15}}{7} + \frac{x_{25}}{7} \ge 1$$

$$\frac{x_{22}}{15} + \frac{x_{32}}{5} \ge 1$$

$$x_{13}, x_{15}, x_{22}, x_{23}, x_{25}, x_{32} \ge 0 .$$

Az LP által adott alsó korlát értéke (17,4 óra) élesebb, mit a leghosszabb út kereső algoritmus által meghatározott (14 óra); az optimum értéke 20.

A bemutatott LP modell segítségével a leghosszabb út kereső algoritmus által adott alsó korláton javítani tudunk. Gyakorlatban azonban a modell felírása és megoldása túl sok időt vesz igénybe ahhoz, hogy hatékony legyen abban az esetben, ha minden részfeladatra alkalmazzuk. A leghosszabb út kereső algoritmus viszont hiába gyors, nagyobb méretű feladatok esetén a gyökér közelében nem ad kellően éles alsó korlátot, így mélyre kell mennünk a kereső fában ahhoz, hogy a részfeladatokat alsó korlát szerint eldobhassuk, ez viszont sok új részfeladat generálását jelenti, ami csökkenti az algoritmus hatékonyságát. A célunk egy olyan módszer kidolgozása volt, amely egyesíti a két alsó korlát meghatározás előnyeit, tehát jó alsó korlátot ad és emellett gyors is. Ezt úgy érhetjük el, hogy az LP modell felépítése közben megbecsüljük, hogy az aktuális részfeladat alsó korlátja változhat-e a szülő részfeladathoz viszonyítva és csak akkor oldjuk meg az LP modellt, ha szükséges.

Az LP modell segítségével meg tudjuk határozni minden egyes berendezéshez a működésének befejezési idejét a relaxált modellben, amit eltárolhatunk az aktuális részfeladatban. Ezen idők meghatározásához bevezetünk egy változót minden berendezéshez (x_i, $i=1,2,...,n$) és módosítjuk az LP modell első feltételét az alábbiak szerint.

$$c_i + \sum_{j \in N} x_{ij} = x_i \leq X \quad (i=1,2,...,n)$$

Jelölje a szülő részfeladathoz tartozó c_i értékeket c_i^* ($i=1,2,...,n$), a várható befejezési időket x_i^* ($i=1,2,...,n$), az alsó korlátot X^*, valamint x_i^b a becsült x_i értékét. A c_i értékek definíciójából következik, hogy a c_i soha nem lesz kisebb, mint c_i^*, tehát két esetet kell megkülönböztetnünk.

1. Ha egy részfeladatnál az i berendezéshez tartozó c_i érték nem változik ($c_i = c_i^*$) a szülő részfeladathoz képest, akkor az azt jelenti, hogy a berendezéssel kapcsolatos (közvetlen vagy közvetett) döntés nem történt, így van olyan megoldása az LP feladatnak, ahol a berendezés várható befejezési ideje x_i sem fog változni, becsülhetjük ezzel az értékkel ($x_i^b = x_i^*$).
2. Ha c_i értéke nő valamely i berendezésre ($c_i > c_i^*$), akkor egy lehetséges (nem feltétlenül optimális) megoldása az LP modellnek, ha x_i értéke is ugyanilyen mértékben nő és a többi érték nem változik, azaz ezzel az értékkel becsülhetjük x_i-t felülről ($x_i^b = x_i^* + c_i - c_i^*$). Ha egyszerre több berendezésnél is változik c_i értéke, akkor mindegyiknél ugyanezt a becslési módszert alkalmazzuk.

Abban az esetben, ha minden i berendezésre x_i becsült értéke kisebb, mint a szülő alsó korlátja ($X^* \geq x_i^b$, $i=1,2,...,n$), akkor létezik olyan megoldása a modellnek, amelyben minden x_i kisebb vagy egyenlő, mint az általunk megbecsült érték úgy, hogy közben az LP modell optimumának értéke nem változik, tehát nem kell az LP modellt megoldani.

Természetesen a becslési eljáráshoz minden részfeladatnál az alsó korlát mellett el kell tárolnunk még a c_i paraméterek és az x_i változók értékeit is minden berendezésre ($i = 1,2,...,n$). Abban az esetben, ha nem hívtunk LP-t, akkor a tényleges értékek helyett a becsült x_i értékeket (x_i^b) tároljuk el.

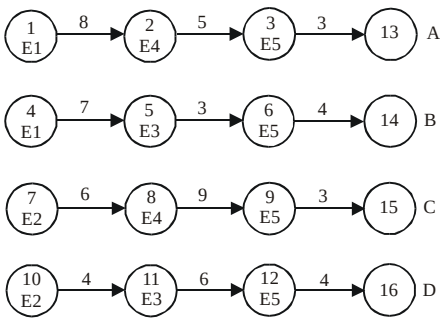
8. Számítógépes megvalósítás

Az alapalgoritmust, a kiterjesztését és gyorsításokat C++ nyelven valósítottam meg és Microsoft Visual C++ 6.0 környezetben fordítottam le. A program parancssorból működik és funkciói kapcsolókon keresztül érhetőek el. A bemenet egy szöveges formátumú fájl, amely tartalmazza a receptet; a kimenet szintén egy szöveges formátumú fájl, amelyben a megoldás értéke, a berendezések működési sorrendjei a megoldáshoz szükséges idő, az iterációk száma és ha volt, akkor az LP hívások száma található.

A futtatásokat a következő számítógépen végeztem el: Intel Celeron 1.2GHz (256KB), 128MB RAM, 20GB 5400RPM IDE HDD, PCI Desktop (3x4), Intel 810e, 48X CD-ROM, Intel 10/100 Ethernet, Windows 98SE. Az alapalgoritmus, a kiterjesztés és a gyorsítások hatékonyságát egy szakirodalomból vett példán keresztül mutatom be.

8. példa

A példát Voudouris és Grossmann (1996) mutatta be. Ebben a példában öt berendezés (E1, E2, E3, E4 és E5) áll rendelkezésre négy termék előállításához. A termékek receptjei a 38. ábrán a recept-gráffal adottak.



38. ábra: Recept-gráf a 8. példához, ha minden termékből egy batch kell.

8.1. Alapalgoritmus

Az alapalgoritmust a 8. példára változó batch számmal futtattam. Ezek a futási eredmények a 4. táblázatban láthatóak. Több mint 11 batch-re a program belátható időn belül (néhány óra) nem futott le.

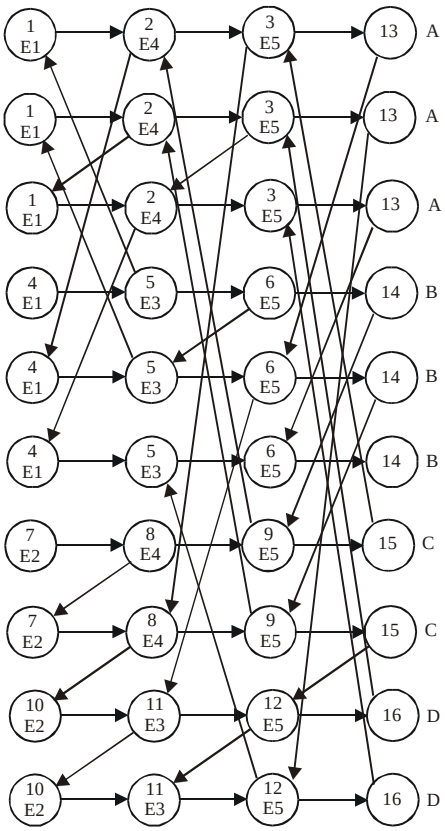
4. táblázat: Futási eredmények a 8. példához az alapalgoritmus esetén

Batch-ek száma				Futási idő (másodperc)	
A	B	C	D		
1	1	1	1	0,06	
2	1	1	1	0,06	
2	2	1	1	0,06	
2	2	2	1	1,81	
2	2	2	2	11,43	
3	2	2	2	26,86	
3	3	2	2	8,62	*
3	3	3	2	4809,66	

*Voudouris és Grossmann (1996) által megoldott példa

A 4. táblázatból észrevehető, hogy a futási idő nincs mindig arányban a batch-ek számával, mivel a batch-ek száma olyan hatással lehet a feladat struktúrájára, hogy egy kisebb méretű feladat bonyolultabb lehet egy nagyobbnál.

Voudouris és Grossmann (1996) által publikált 10 batch-es példa eredményének értéke megegyezik az alapalgoritmus által generálttal. Az alapalgoritmus 8,62 másodperc alatt futott le egy Intel Celeron 1,2 GHz számítógépen, összehasonlítva Voudouris és Grossmann módszerével, a 293 másodpercig futó GAMS 2.25/Sciconic 2.11-t használó IBM/R6000/Power 530 munkaállomással. Figyelembe véve a két számítógép közötti teljesítménykülönbséget is látható, hogy az algoritmus hatékonysága bármiféle gyorsítás nélkül is összemérhető egy általános célú MILP megoldóval. Az algoritmus által generált optimális megoldás a 39. ábrán látható.



39. ábra: A 8. példa egy optimális megoldása 10 batch esetén.

8.2. Több batch kezelése

Az alapalgoritmust a 8. példa segítségével hasonlítottam össze azzal az algoritmussal, amelyben a batch-ek kezelése megoldott volt (6. fejezet). A futási idők változása az alapalgoritmushoz képest az 5. táblázatban láthatóak, az iterációk számának változása pedig a 6. táblázatban.

Látható, hogy amikor minden termékből csak egy batch-et kellett előállítani, akkor a két futtatás között nem volt különbség (mint ahogy az várható is volt). Amikor az A termékből a szükséges mennyiséget megdupláztuk, akkor az iterációk száma az alapalgoritmusnál csökkent, a batch kezelésnél nőtt az alapalgoritmushoz képest (ez a feladat speciális tulajdonságaiból adódik). Általában viszont az figyelhető meg, hogy a batch-ek számának növelésével a gyorsítás lineárisnál nagyobb mértékben növekszik. Az iterációk számát figyelve a gyorsítási nagyobb mértékű, ez abból adódik, hogy a gráfban lévő segéd-élek adminisztrációja plusz időt jelent a megvalósításban.

5. táblázat: A futási idők változása batch kezelés esetén, összehasonlítva az alapalgoritmussal.

Batch-ek száma				Alapalgoritmus	Batch kezelés	Gyorsítási arány
A	B	C	D	CPU idő (mp)	CPU idő (mp)	
1	1	1	1	0,06	0,06	1,00
2	1	1	1	0,06	0,06	1,00
2	2	1	1	0,06	0,06	1,00
2	2	2	1	1,81	0,55	3,29
2	2	2	2	11,43	1,38	8,26
3	2	2	2	26,86	2,58	10,41
3	3	2	2	8,62	0,49	17,59
3	3	3	2	4809,66	17,36	277,05
3	3	3	3	-	57,29	-
4	3	3	3	-	102,11	-
4	4	3	3	-	12,96	-
4	4	4	3	-	781,04	-
4	4	4	4	-	2315,00	-

6. táblázat: Az iterációk számának változása batch kezelés esetén, összehasonlítva az alapalgoritmussal.

Batch-ek száma				Alapalgoritmus	Batch kezelés	Gyorsítási arány
A	B	C	D	Iterációk száma	Iterációk száma	
1	1	1	1	43	43	1,00
2	1	1	1	41	53	0,77
2	2	1	1	48	20	2,40
2	2	2	1	1167	248	4,71
2	2	2	2	5730	499	11,48
3	2	2	2	12420	739	16,81
3	3	2	2	3476	90	38,62
3	3	3	2	1223995	3512	348,52
3	3	3	3	-	9158	-
4	3	3	3	-	14872	-
4	4	3	3	-	1126	-
4	4	4	3	-	76483	-
4	4	4	4	-	186179	-

8.3. Előzetes körfelismerés

Az előzetes körfelismeréssel gyorsított algoritmust az alapalgoritmussal hasonlítottam össze. Az összehasonlításhoz a 8. példát vettem alapul változó batch számmal. A futási idők változása az alapalgoritmushoz képest a 7. táblázatban láthatóak, az iterációk számának változása pedig a 8. táblázatban.

7. táblázat: A futási idők változása előzetes körfelismerő algoritmus használata esetén, összehasonlítva az alapalgoritmussal.

Batch-ek száma				Alapalgoritmus	Előzetes körfelismerés	Gyorsítási arány
A	B	C	D	CPU idő (mp)	CPU idő (mp)	
1	1	1	1	0,06	0,06	1,00
2	1	1	1	0,06	0,06	1,00
2	2	1	1	0,06	0,06	1,00
2	2	2	1	1,81	1,65	1,10
2	2	2	2	11,43	9,67	1,18
3	2	2	2	26,86	22,41	1,20
3	3	2	2	8,62	7,25	1,19
3	3	3	2	4809,66	3546,71	1,36

8. táblázat: Az iterációk számának változása előzetes körfelismerő algoritmus használata esetén, összehasonlítva az alapalgoritmussal.

Batch-ek száma				Alapalgoritmus	Előzetes körfelismerés	Gyorsítási arány
A	B	C	D	Iterációk száma	Iterációk száma	
1	1	1	1	43	43	1,00
2	1	1	1	41	41	1,00
2	2	1	1	48	48	1,00
2	2	2	1	1167	1167	1,00
2	2	2	2	5730	5698	1,01
3	2	2	2	12420	11940	1,04
3	3	2	2	3476	3332	1,04
3	3	3	2	1223995	1192891	1,03

A futási eredmények nem igazolták az előzetes várakozásainkat, ugyanis még nem volt nagy mértékű a gyorsítás. A további vizsgálatokból kiderült, hogy a gráf nem volt eléggé összekötött, ezért a gyorsítás hatását batch kezeléssel együtt is megvizsgáltuk, amikor a gráfot segéd-élekkel kiegészítettük. A futtatási eredmények a 9. és a 10. táblázatban láthatók.

Ebben az esetben már látszik, hogy az előzetes körfelismerő algoritmus segítségével további, időben körülbelül kétszeres gyorsítást tudtunk elérni nagyobb méretű, több batch-et tartalmazó feladatok esetén. Érdemes azt is megjegyezni, hogy az eredeti Voudouris és Grossmann (1994) által 293 másodperc alatt megoldott feladatot a két gyorsítás együttes használatával 0,33 másodperc alatt képes az algoritmus megoldani. Ha figyelembe vesszük a PC

nagyobb számítási teljesítményét, akkor is nagyságrendbeli gyorsítást jelent úgy, hogy az optimalitást garantálni lehet, ellentétben a másik megközelítéssel.

9. táblázat: A futási idők változása batch kezelés és előzetes körfelismerés együttes használta esetén, összehasonlítva az alapalgoritmussal és a batch kezeléssel.

Batch-ek száma				Alapalgoritmus	Batch kezelés	Batch kezelés és előzetes körfelismerés
A	B	C	D	CPU idő (mp)	CPU idő (mp)	CPU idő (mp)
1	1	1	1	0,06	0,06	0,06
2	1	1	1	0,06	0,06	0,06
2	2	1	1	0,06	0,06	0,06
2	2	2	1	1,81	0,55	0,44
2	2	2	2	11,43	1,38	0,99
3	2	2	2	26,86	2,58	1,81
3	3	2	2	8,62	0,49	0,33
3	3	3	2	4809,66	17,36	11,09
3	3	3	3	-	57,29	34,88
4	3	3	3	-	102,11	57,07
4	4	3	3	-	12,96	6,76
4	4	4	3	-	781,04	401,84
4	4	4	4	-	2315,00	1158,54

10. táblázat: Az iterációk számának változása batch kezelés és előzetes körfelismerés együttes használata esetén, összehasonlítva az alapalgoritmussal és a batch kezeléssel.

Batch-ek száma				Alapalgoritmus	Batch kezelés	Batch kezelés és előzetes körfelismerés
A	B	C	D	Iterációk száma	Iterációk száma	Iterációk száma
1	1	1	1	43	43	43
2	1	1	1	41	53	53
2	2	1	1	48	20	20
2	2	2	1	1167	248	243
2	2	2	2	5730	499	473
3	2	2	2	12420	739	689
3	3	2	2	3476	90	85
3	3	3	2	1223995	3512	3192
3	3	3	3	-	9158	8452
4	3	3	3	-	14872	12529
4	4	3	3	-	1126	1058
4	4	4	3	-	76483	67549
4	4	4	4	-	186179	168094

8.4. LP modell a leghosszabb út keresés javításához

A leghosszabb út keresés javításához kidolgozott LP modell hatékonyságát a batch kezeléssel kiterjesztett és az előzetes körfelismeréssel gyorsított algoritmuson vizsgáltuk. Az LP modell minden lépésben megoldó algoritmust egyszerre teszteltük azzal, amelyben a hívások száma csökkentettük. Az LP feladat megoldásához Fábián (1985) LINX megoldóját vettem alapul. A 8. példához tartozó futási idők a 11. táblázatban, az iterációk száma a 12. táblázatban, valamint az LP hívások száma a 13. táblázatban láthatóak. A 14. táblázatban láthatóak a recept-gráfra adott alsó korlátok az optimum értékével összehasonlítva.

11. táblázat: A futási idők változása LP modell használata esetén a már gyorsított algoritmussal összehasonlítva.

Batch-ek száma				Leghosszabb út keresés	LP modell	Csökkentett LP hívás szám
A	B	C	D	CPU idő (mp)	CPU idő (mp)	CPU idő (mp)
1	1	1	1	0,06	0,11	0,11
2	1	1	1	0,06	0,22	0,22
2	2	1	1	0,06	0,11	0,06
2	2	2	1	0,44	1,15	0,99
2	2	2	2	0,99	2,47	2,08
3	2	2	2	1,81	4,23	3,62
3	3	2	2	0,33	2,14	1,81
3	3	3	2	11,09	22,74	18,62
3	3	3	3	34,88	65,80	55,80
4	3	3	3	57,07	107,65	91,45
4	4	3	3	6,76	40,59	36,03
4	4	4	3	401,84	644,72	565,40
4	4	4	4	1158,54	1877,90	1603,28

A 8. példa megoldása során azt vehetjük észre, hogy az LP hívások számának csökkentése az iterációk számát nem változtatja meg (mint ahogy ez várható is volt). A futási időben a hívásszám csökkentése nem jelent nagy mértékű gyorsulást, mivel a c_i konstansok meghatározásához be kellett járni az egész gráfot, tehát az LP modell meghatározásának nagy része ebben az esetben is megtörténik, csak a modell konkrét megoldásának idejével csökken a megoldási idő. Jelenleg is kidolgozás alatt van egy olyan számítógépes gráf leírás, amely a c_i konstansok meghatározását meggyorsíthatja.

12. táblázat: Az iterációk számának változása LP modell használata esetén a már gyorsított algoritmussal összehasonlítva.

Batch-ek száma				Leghosszabb út keresés	LP modell	Csökkentett LP hívás szám
A	B	C	D	Iterációk száma	Iterációk száma	Iterációk száma
1	1	1	1	43	25	25
2	1	1	1	53	50	50
2	2	1	1	20	20	20
2	2	2	1	243	218	218
2	2	2	2	473	404	404
3	2	2	2	689	626	626
3	3	2	2	85	285	285
3	3	3	2	3192	2832	2832
3	3	3	3	8452	7359	7359
4	3	3	3	12529	11310	11310
4	4	3	3	1058	3841	3841
4	4	4	3	67549	57258	57258
4	4	4	4	168094	150658	150658

13. táblázat: Az LP hívások számának változása a csökkentett LP hívás szám esetén.

Batch-ek száma				LP modell	Csökkentett LP hívás szám	Csökkentési arány
A	B	C	D	Hívások száma	Hívások száma	
1	1	1	1	40	19	2,11
2	1	1	1	81	43	1,88
2	2	1	1	36	21	1,71
2	2	2	1	355	200	1,78
2	2	2	2	680	355	1,92
3	2	2	2	1069	600	1,78
3	3	2	2	488	294	1,66
3	3	3	2	4844	2785	1,74
3	3	3	3	12812	7518	1,70
4	3	3	3	19653	11342	1,73
4	4	3	3	6626	4140	1,60
4	4	4	3	99610	60072	1,66
4	4	4	4	266949	161424	1,65

Az eredmények vizsgálatával arra a következtetésre juthatunk, hogy az LP modell bevezetése nem gyorsítja az algoritmust, hanem inkább lassítja azt. Az iterációk száma ugyan csökken (12. táblázat) és az alsó korlát javul (14. táblázat), de a futási idő ezzel szemben növekszik (11. táblázat). Ez a megállapítás nem minden feladat esetében igaz. Nagyobb, összetettebb példáknál a leghosszabb út kereső algoritmus nem biztosít elegendően éles alsó korlátot, emiatt nem lehet időben észre venni, ha a keresőfában rossz irányba indul az algoritmus. Ha ez a

tévedés a fa gyökeréhez közel történik meg (itt gyenge az alsó korlát), akkor sokáig tart, mire az algoritmus visszatalál a megfelelő úthoz. A másik problémát nem az optimum megtalálása jelenti, hanem a megoldás optimalitásának bizonyítása, azaz a többi részfeladat eldobása. Ha a leghosszabb út kereső algoritmust használjuk, akkor a keresőfában nagyon mélyre kell lemenni ahhoz, hogy az alsó korlát elérje vagy meghaladja az addig megtalált legjobb megoldás értékét. Egy konkrét ipari alkalmazási feladat leegyszerűsített változatán mutatjuk be az LP modell használhatóságát.

14. táblázat: Az alsó korlát értéke a recept-gráfra, összehasonlítva az optimum értékével.

Batch-ek száma				Alsó korlát a recept-gráfra		Optimum
A	B	C	D	Leghosszabb út keresés	LP modell	
1	1	1	1	18	24	25
2	1	1	1	24	27	31
2	2	1	1	24	31	37
2	2	2	1	27	34	39
2	2	2	2	27	38	41
3	2	2	2	32	41	46
3	3	2	2	32	45	52
3	3	3	2	36	48	55
3	3	3	3	36	52	57
4	3	3	3	40	55	62
4	4	3	3	40	60	67
4	4	4	3	45	62	71
4	4	4	4	45	66	73

9. példa

Tizenkilenc berendezés (E1-től E19-ig) áll rendelkezésre tíz termék (A-tól J-ig) előállításához. A termékek receptjei a 15. és a 16. táblázatban olvashatók. A váltási idő 60 perc az E1, E2, E3 és E4 berendezésekhez, 90 perc az E5, E6, E7, E8, E9, E10, E11, E12, E13, E14 és E15 berendezésekhez és nulla az összes többi berendezéshez (E16, E17, E18 és E19). A szükséges termékmennyiség előállításához tartozó batch-ek száma a 17. táblázatban látható. A feladatot futtattuk a leghosszabb út kereső algoritmust illetve a csökkentett LP hívások számának módszerét használva is. A feladat megoldásánál az eddigiektől eltérően a berendezéseket a leterheltség sorrendjében ütemeztük.

15. táblázat: Az A, B, C, D és E termékek receptje a 9. példához.

Taszk	A termék		B termék		C termék		D termék		E termék	
	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)
1	E1	300	E2	240	E2	240	E1	300	E2	240
			E3	120	E3	120	E3	120	E4	240
					E4	240				
2	E5	60	E6	120	E6	120	E6	120	E5	60
	E6	120	E7	90	E8	60	E7	90	E6	120
	E8	60	E8	60	E10	120	E8	60	E8	60
	E10	120			E12	60	E9	90	E9	90
	E12	60			E14	120	E11	90	E10	120
	E14	120			E15	60			E12	60
	E15	60							E13	90
									E14	120
									E15	60
3	E16	720	E19	840	E17	540	E16	720	E17	540
	E17	540			E18	720	E17	540	E18	720

16. táblázat: Az F, G, H, I és J termékek receptje a 9. példához.

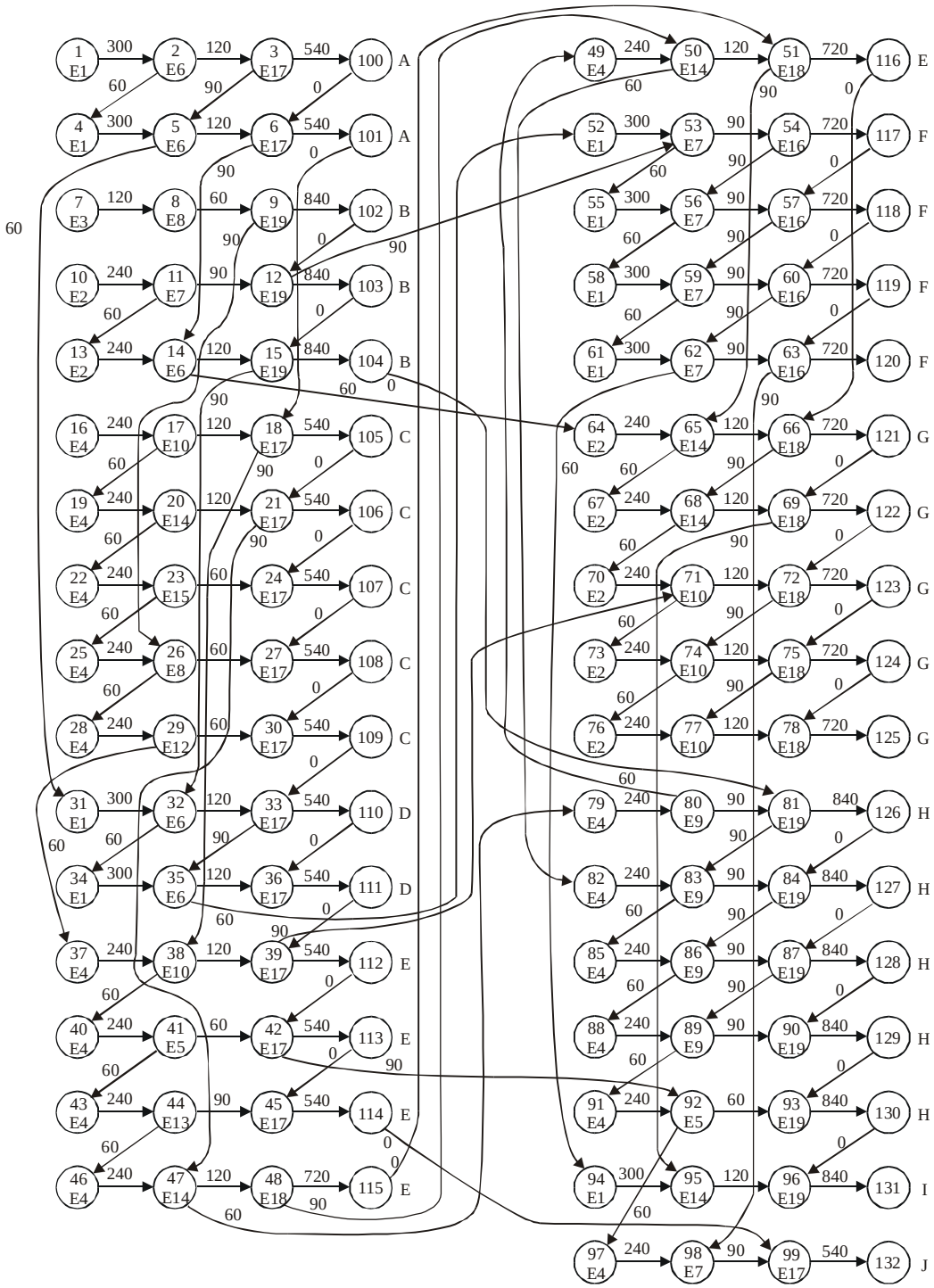
Taszk	F termék		G termék		H termék		I termék		J termék	
	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)	Beren-dezés	Idő (perc)
1	E1	300	E2	240	E4	240	E1	300	E4	240
	E3	120	E3	120			E3	120		
2	E5	60	E5	60	E5	60	E10	120	E7	90
	E7	90	E10	120	E7	90	E11	90	E9	90
	E9	90	E11	90	E9	90	E12	60		
	E11	90	E12	60	E11	90	E13	90		
	E13	90	E13	90	E13	90	E14	120		
	E15	60	E14	120						
			E15	60						
3	E16	720	E16	720	E19	840	E19	840	E17	540
	E17	540	E17	540						
			E18	720						
			E19	840						

17. táblázat: A batch-ek száma a 9. példához.

Termék	A	B	C	D	E	F	G	H	I	J
Batch-ek száma	2	3	5	2	5	4	5	5	1	1

A példánál a leghosszabb út kereső algoritmus a recept-gráfra 2100 perces alsó korlátot adott, míg az LP modell 7740 percet, ami egyben megegyezik az optimum értékével. A leghosszabb út kereső algoritmus használata esetén a program 11,04 másodperc alatt talált egy 8580 perces megoldást, viszont 2 nap futás sem volt elegendő egy jobb megoldás megtalálásához a használt számítógépen (Intel Celeron 1,2 GHz). Ugyanezen feladatnál a csökkentett LP hívások módszerét alkalmazva a program ugyanennyi idő alatt (11,04 másodperc) még nem talált megoldást, viszont 30,76 másodperc alatt megtalálta az optimumot

és bizonyította annak globalitását (kizárta az összes többi részfeladatot). A feladat egy optimális megoldásának ütemezési-gráfja a 40. ábrán látható.



40. ábra: A 9. példa egy optimális megoldásának ütemezési-gráfja.

9. Összefoglalás

A disszertáció elején bemutattam a szakaszos működésű rendszerek ütemezési feladatait és a megoldásához kifejlesztett S-gráf leírást, valamint definiáltam speciális S-gráfokat az ütemezési feladat matematikai leírásához. Bevezettem a recept-gráf fogalmát a betáplálási-sorrend gráffal együtt az ütemezési feladatok receptjének matematikai leírásához. Bevezettem továbbá a komponens-gráfot és meghatároztam négy feltételt, amelyek segítségével definiálni tudtam az ütemezési-gráfot, amely a berendezés-taszk hozzárendeléssel együtt a feladat egy konkrét megoldását határozza meg.

A disszertáció második részében bemutattam egy az S-gráf leíráson alapuló alapalgoritmust ipari termelő rendszerek ütemezéséhez. Az ütemezési feladatok MILP vagy MINLP modelljének generálása és megoldása helyett egy speciális módszert mutattam be, amely hatékony gráf algoritmusokon alapul (leghosszabb út keresés és körkeresés algoritmus) továbbá kihasználja az ütemezési feladatok speciális tulajdonságait. Az alapalgoritmus és az S-gráf leírás rugalmassága jó alapot biztosít speciális algoritmusok készítésére, speciális ütemezési feladatok megoldásához. Ilyen lehet például a rugalmas recepttel rendelkező termékeket tartalmazó feladatok (Romero és társai, 2003) vagy a komplex recepttel rendelkező termékeket tartalmazó feladatok, de ki lehet terjeszteni az S-gráf leírást véges méretű köztes tárolások esetére is (Romero és társai, 2003) illetve integrálni más típusú feladatokkal (Adonyi és társai, 2003).

Az algoritmust kiterjesztettem arra az estre, amikor egy termékből több batch-et kellett gyártani. A feladatot az alapalgoritmus is képes megoldani, de feltétetlek bevezetésével a keresési tér jelentősen lecsökkenthető és az algoritmus hatékonysága növelhető. A feltételeket az S-gráfban élek felvételével, az úgy nevezett segéd-élek segítségével reprezentáltam.

Ezek után bemutattam két az algoritmushoz általánosan használható gyorsítást, amelyek segítségével az algoritmus hatékonysága nagy mértékben megnőtt. Az első gyorsítás a keresési teret csökkenti figyelembe véve egy

egyszerű, az időzítésekkel kapcsolatos szabályt. Ezzel a módszerrel már a kialakulása előtt detektálni lehet egy kört és el lehet kerülni azon részfeladatok vizsgálatát, amelyek biztosan nem vezetnek megvalósítható megoldáshoz. A második gyorsítás egy alsó korlát meghatározó LP modell felírását jelenti, amely a leghosszabb út kereső algoritmus értékét pontosítja, valamint az LP hívások számát csökkentettem egy becslő eljárás segítségével.

A disszertáció végén bemutattam az alapalgoritmus, a kiterjesztett algoritmus és a beépített gyorsítások hatékonyságát egy a szakirodalomból származó példán, ahol nagyságrendbeli gyorsulást tudtam elérni. Továbbá egy konkrét ipari alkalmazás leegyszerűsített feladatát oldottam meg az algoritmussal megmutatva, hogy a megoldható feladatok nagysága az eddig a szakirodalomban használt módszerekénél nagyobb úgy, hogy a globális optimumot továbbra is garantálni lehet.

10. Jelölésjegyzék

n	berendezések száma
N	csomópontok halmaza
N_t	taszk-csomópontok halmaza
N_p	termék-csomóponttok halmaza
N_f	betáplálási-sorrend csomóponttok halmaza
N_i	az i berendezéssel végrehajtható taszk-csomópontok halmaza ($i = 1,2,...,n$)
M_i	a megoldásban az i berendezéshez rendelt taszk-csomópontok halmaza ($i = 1,2,...,n$)
A_1	recept-élek halmaza
A_2	ütemezési-élek halmaza
$c(i, j)$	az (i, j) él súlya ($(i, j) \in A_1 \cup A_2$)
$G(N, A_1, A_2)$	S-gráf
$G(N, A_1, \varnothing)$	recept-gráf
$G'(N, A_1, A_2)$	ütemezési-gráf
$G_i'(N_i', A_{1i}, A_{2i})$	az i berendezés komponens-gráfja ($i = 1,2,...,n$)
S_j	a j taszkot végrehajtható berendezések halmaza ($j = 1,2,...,\lvert N_t \rvert$)
$\overline{M}_i$	be nem ütemezett, csak az i berendezéssel végrehajtható csomópontok halmaza ($i = 1,2,...,n$)
$\overline{N}$	be nem ütemezett, több berendezéssel végrehajtható csomópontok halmaza
L_j	a j csomópontba vezető leghosszabb út hossza ($j = 1,2,...,\lvert N_t \rvert$)
t_{ij}	az i berendezés működési ideje a j taszkon ($i = 1,2,...,n$, $j = 1,2,...,\lvert N_t \rvert$)
c_i	alsó korlát az i berendezés munkájának befejezési idejére ($i = 1,2,...,n$)

c_i^*	a szülő részfeladathoz tartozó c_i érték ($i = 1,2,...,n$)
x_{ij}	(változó) az i berendezés működési ideje a j taszkban ($i = 1,2,...,n,\ j \in \overline{N}$)
x_i	(változó) az i berendezés munkájának várható befejezési ideje ($i = 1,2,...,n$)
x_i^*	a szülő részfeladathoz tartozó x_i érték ($i = 1,2,...,n$)
x_i^b	becsült x_i érték ($i = 1,2,...,n$)
X	(változó) alsó korlát a részfeladatból elérhető megoldások befejezési idejére

11. Irodalomjegyzék

Adams, J., E. Balas, D. Zawack, 1988, The shifting bottleneck procedure for job shop scheduling, *Management Science*, **34**, 391-401.

Adonyi, R., J. Romero, L. Puigjaner, and F. Friedler, 2003, Incorporating heat integration in batch process scheduling, Applied Thermal Engineering, **23**, 1743-1762.

Carlier, J., E. Pinson, 1989, An algorithm for solving the job-shop problem, *Management Science*, **35**, 164-176.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, 1997, Introduction to algorithms, The MIT Press.

Elkamel, A., 1993, Scheduling of Process Operations using Mathematical Programming Techniques, PhD Thesis, Purdue University.

Fábián, CS., 1985, LINX, lineáris programozási programrendszer, Felhasználói kézikönyv, Budapest, 1-28.

Glover, F., 1975, Improved linear integer programming formulations of nonlinear integer problems, *Man. Sci.*, **22**(**4**), 455-460.

Graells, M., A. Espuña, L. Puigjaner, 1996, Sequencing intermediate products: A practical solution for multipurpose production scheduling, *Computers chem. Engng*., **20S**, S1137-S1142.

Graells, M., J. Cantón, J., B. Peschaud, L. Puigjaner, 1998, General Approach and Tool for the Scheduling of Complex Production Systems, *Computers & Chemical Engineering*, **22S**, S395-S402.

Grossmann, I. E., Sargetnt, R. W. H., 1979, Optimum design of multipurpose chemical plants, *Ind. Eng. Chem. Proc. Des. Dev.*, **18**, 343-348.

Hasebe, S., Taniguchi, S., Hashimoto, I.,1996, Automatic Adjustment of Crossover Method in the Scheduling Using Genetic Algorithm, *Kagaku Kogaku Ronsbunshu*, **22,** 1039-1045.

Ierapetritou, M. G., Floudas, C. A., 1998, Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes, *Ind. Eng. Chem. Res.*, **37**, 4341-4359.

Knopf, F. C., Okos, M. R., Reklaitis, G., V., 1982, Optimal design of batch/semicontinuous processes, *Ind. Eng. Chem. Proc. Des. Dev.*, **21**, 79-86.

Kondili, E., Pantelides C. C., Sargent, R. W. H. 1993, A general algorithm for short-term scheduling of batch operations-I. MILP formulation, *Comp. Chem. Eng.*, **17**(**2**), 211-227.

Kudva, G., Elkamel, A., Pekny, J. F., Reklaitis, G. V., 1994, Heuristic Algorithm for Scheduling Batch and Semicontinuous Plants with Production Deadlines, Intermediate Storage Limitations and Equipment Changeover Costs, *Computer chem. Engng. Symp. Ser.*, **18**, 859-875.

Lee, H. K., Lee, I. B., 1996, A synthesis of multiproduct batch plants considering both in-phase and out-phase modes, *Comp. Chem. Eng.*, **20S**, S195-S200.

Mockus L., Reklaitis, G. V., 1997, Mathematical Programming Formulation for Scheduling of Batch Operations Based on Nonuniform Time Discretization, *Comput. & Chem. Engr.*, **21**, 1147-1156.

Murakami, Y., Uchiyama, H., Hasebe, S., Hashimoto, I., 1997, Application of Repetitive SA Method to Scheduling Problems of Chemical Processes, *Computers chem. Engng.*, **S21**, S1087-S1092.

Pantelides C. C., 1994, Unified Frameworks for optimal process planning and scheduling, *Proc. Second Conference on Foundations of Computer-Aided Operations (FOCAPO II)*, 235-274.

Pekny, J., G. Reklaitis, 1998, Towards the Convergence of Theory and Practice: A Technology Guide for Scheduling /Planning Methodology, *Foundations of Computer-Aided Process Operations. AIChE Symposium Series No.320.* (Eds. Joseph F. Pekny and Gary E. Blau), American Institute of Chemical Engineers (AIChE), New York, **94**, 91-111.

Pinto, J. M., I. E. Grossmann, 1995, A Continuous Time Mixed Integer Linear Programming Model for Short-Term Scheduling of Multistage Batch Plants, *Ind. Eng. Chem. Res.*, **34**, 3037-3051

Puigjaner, L., 1999, Handling the increasing complexity of detailed batch process simulation and optimisation, *Computers & Chemical Engineering*, **23S** 1354), S929-S943.

Ravemark, D. E., Rippin, D. W. T., 1998, Optimal design of a multi-product batch plant, *Comp. Chem. Eng.*, **22**, 177-183.

Reklaitis, G. V., 1981, Review of Scheduling of Process Operations, *AIChE Annual Meeting*, New Orleans, Nov.

Romero, J., A. Espuna, F. Friedler, and L. Puigjaner, 2003 A New Framework for Batch Process Optimization Using the Flexible Recipe, Industrial and Engineering Chemistry Research, **42**(2), 370-379.

Sahinidis, N. V., Grossmann, I. E., Fornari, R. E., Chathrathi, M., 1991, Optimisation Model for Long-Range Planning in the Chemical Industry, *Computer chem. Engng.*, **15**, 255-272.

Sanmartí, E., A. Espuña, L. Puigjaner, 1996, An MILP formulation for the production scheduling of multipurpose batch chemical plants, *7th Mediterranean Congress of Chemical Engineering*, Barcelona (Spain).

Sanmartí, E., F. Friedler, L. Puigjaner, 1998, Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation, *Computers chem. Engng*., Vol. **22,** Suppl., pp. S847-S850.

Sanmartí, E., T. Holczinger, F. Friedler, L. Puigjaner, 2002, Combinatorial Framework for Effective Scheduling of Multipurpose Batch Plants, AIChE Journal, **48**(**11**), 2557-2570.

Schilling, G., Pantelides, C. C., 1996, A Simple Continuous Time Process Scheduling Formulation and a Novel Solution Algorithm, *Computers chem. Engng.*, **S20**, S1221-S1226.

Shah, N., Pantelides, C. C., Sargent, R. W. H., 1993, A General Algorithms for Short-Term Scheduling of Batch Operations – 2. Computational Issues, *Computers chem. Engng.*, **17**, 229-244.

Sparrow, R. E., Forder, G. J., Rippin, D. W. T., 1975, The choice of equipment sizes for multiproduct batch plants – heuristic vs. branch and bound, *Ind. Eng. Chem. Proc. Des. Dev.*, **14**, 197-203.

Suhami, I., Mah, R. S. H., 1982, Optimal design of multipurpose batch plants, *Ind. Eng. Chem. Proc. Des. Dev.*, **21**, 94-100.

Tan, S. T., Mah, R. S. H., 1998, Evolutionary design of noncontinuous plants, *Comp. Chem. Eng.*, **22**(**1-2**), 69-85.

Voudouris, V. T. and I. E. Grossmann, 1996, MILP model for scheduling and design of a special class of multipurpose batch plants, *Computers chem. Engng.*, Vol. **20,** No. 11, pp. 1335-1360.

Yee, K. L., Shah, N., 1998, Improving the Efficiency of Discrete-Time Scheduling Formulations, *Computers chem. Engng.*, **S22**, S403-S410.

Zhang X., 1995, Algorithms for optimal scheduling using nonlinear models, PhD Thesis, University of London.

Zhang X., Sargent, R. W. H., 1994, The optimal operation of mixed production facilities – a general formulation and some solution approaches for the solution, *Proc. of 5th Intl. Symp. on Process Systems Engineering*, Kyongju, Korea, 171-177.

Új tudományos eredmények összefoglalása

1. Tézis: Az ütemezési feladat kombinatorikus tulajdonságait figyelembe véve olyan egységes matematikai modellt dolgoztam ki S-gráf alkalmazásával a feladat leírásához, amely hatékony megoldó algoritmusok kidolgozását teszi lehetővé.
   1.1. Bevezettem a „recept-gráf” és a „betáplálási-sorrend gráf” fogalmát, amelyek az ütemezési feladatok leírásában szereplő strukturális tulajdonságokat a megoldó algoritmusok számára célszerű módon tartalmazza.
   1.2. Meghatároztam két olyan strukturális feltételt, amelyek teljesülése szükséges és elegendő feltétele annak, hogy egy S-gráf az ütemezési feladat megoldását reprezentálja.
   1.3. Meghatároztam két olyan strukturális feltételt, amelynek teljesülésével az ütemezési feladat megoldását reprezentáló S-gráf minimális abban az értelemben, hogy él elhagyásával nem ír le megoldást.

2. Tézis: Gyorsító módszert dolgoztam ki a Sanmartí és társai (2002) által kidolgozott alapalgoritmushoz arra a gyakorlatban fontos esetre, amikor egy termékből több (rendszerint nagyszámú) batch előállítására van szükség.
   2.1. Algoritmikusan szűkítettem a keresési teret segéd-élek bevezetésével, melyek segítségével a batch-ek sorrendje előre meghatározható, és az optimum biztosítható.
   2.2. Megvizsgáltam a batch-ek sorrend megkötésének speciális esetét, amikor további segéd-élek felvételével illetve transzformációjával a 2.1. tézispontban leírt módszerhez képest tovább csökkentettem a keresési teret.

3. Tézis: Két általánosan használható gyorsító eljárást fejlesztettem ki az S-gráf leíráson alapuló B&B ütemezési alapalgoritmushoz, amelyek közül az első a keresési teret csökkenti, a második pedig élesebb alsó korlát meghatározását teszi lehetővé.

3.1. Algoritmust dolgoztam ki, amelynek segítségével egy az alapalgoritmus által generált részfeladatnál a leszármazottak megoldása nélkül fel lehet ismerni, hogy a leszármazottak között nincs megoldás (un. look-ahead gyorsítás).

3.2. Megadtam egy lineáris programozási (LP) modellt, amely az aktuális részfeladat S-gráfjából elérhető ütemezési-gráfokhoz tartozó leghosszabb út hosszára alsó korlátot ad. Ez nagy számú egész változó esetén, amikor kevés egész változó van rögzítve a B&B keresés során, jól használható módszernek bizonyult. Eljárást vezettem be, mellyel az LP modell megoldása előtt meghatározhatjuk, hogy az alsó korlát legfeljebb mennyivel növekedhet a szülő részfeladathoz képest. Ha ez az érték nulla, akkor felesleges az LP modell megoldás erre a részfeladatra.

Major results and summary of accomplishments

1. Thesis: Taking into account the combinatorial properties of the scheduling problems I have developed a mathematical model for the problems that gives good basis to development of efficient algorithms.
   1.1. I have introduced the notations of the “recipe-graph” and the “feed-precedence graph” which represent the structural information of the problem in practical way for the solving algorithms.
   1.2. I have determined two structural conditions which are necessary and sufficient conditions for an S-graph to represent a solution of a scheduling problem.
   1.3. I have determined two structural conditions which guarantee that an S-graph representing a solution cannot represent a solution leaving out any arc of it.

2. Thesis: I have developed an acceleration method for the basic algorithm to the case, when a product has more (usually a big number of) than one batch.
   2.1. I have reduced the searching space algorithmically by defining auxiliary-arcs by which the order of the batches belonging to a product can be preliminarily determined without losing the optimality.
   2.2. I have examined the special cases of the ordering of the batches when I reduced the searching space by the additional auxiliary-arcs mentioned in thesis 2.1.

3. Thesis: I have developed two general purpose acceleration procedures to the basic algorithm, where the first one is reducing the searching space, and the second one can determine sharper lower bounds.
   3.1. I have developed an algorithm that detects whether none of the inherit partial problem of the actual partial problem can be a solution without the generation of them (so called look-ahead acceleration).
   3.2. I have given an LP (linear programming) model, which provides a lower bound for the longest path of the inherited schedule-graphs. This model was a good method when the problem had a big number of integer

variables and only few of them were fixed. I have developed a method to determine the maximum value of the increasing of the lower bound compared to the parent partial problem before solving the LP model. If this value is equal to zero, it is unnecessary to solve the LP model for the actual partial problem.

Publikációs tevékenységem

Referált nemzetközi publikációk:

Romero, J., T. Holczinger, F. Friedler, and L. Puigjaner, 2003, Scheduling Intermediate Storage Multipurpose Batch Plants Using the S-graph, accepted for AIChE Journal.

Holczinger, T., J. Romero, F. Friedler, L. Puigjaner, 2002, Scheduling of Multipurpose Batch Processes with Multiple Batches of the Products, Hungarian Journal of Industrial Chemistry, **30**, 305-312.

Sanmartí, E., T. Holczinger, L. Puigjaner, F. Friedler, 2002, Combinatorial Framework for Effective Scheduling of Multipurpose Batch Plants, AIChE Journal, **48**(11), 2557-2570.

Nemzetközi konferencia előadások:

Romero, J., T. Holczinger, F. Friedler, L. Puigjaner, 2002, Modeling the Scheduling of Common Intermediate Storage Multipurpose Batch Process Using a Graph Theoretical Approach, presented at the AIChE Annual Meeting, Indiana, U.S.A., November 7.

Holczinger, T., I. Heckl, A. C. Kokossis, F. Friedler, 2001, Accelerated contextual optimization for scheduling large-scale HPC liquid factories, presented at the AIChE Annual Meeting, Reno, NV, U.S.A., November 4-9.

Holczinger, T., L. Kalotai, F. Friedler, 2000, Scheduling of Process Systems Comprising Continuous and Batch Operations, presented at the CHISA 2000 (14th International Congress of Chemical and Process Engineering), Praha, Czech Republic, August 27-31.

Holczinger, T., L. Kalotai, F. Friedler, 2000, Scheduling of Process Systems Comprising Continuous and Batch Operations, presented at the 20th Workshop on Chemical Engineering Mathematics, Veszprém, Hungary, July 26-29.

Holczinger, T., F. Friedler, L. T. Fan, 1998, Process Synthesis for Retrofit Design, presented at the CHISA ’98 (13th International Congress of Chemical and Process Engineering), Praha, Czech Republic, August 23-28.

Melléklet

Gyakran használt fogalmak és definíciók

Recept (recipe)

A termékek előállítási módja.

Batch (batch)

Termék előállítási folyamata, mely során adott mennyiségű termék keletkezik.

Taszk (task)

Termék előállításának elemi lépése, mely során adott mennyiségű anyagból adott mennyiségű anyag keletkezik.

Berendezés (equipment unit)

Eszköz a taszkok végrehajtásához.

Több terméket előállító rendszer (multiproduct plant)

Olyan termelő rendszer, ahol a termékeket ugyanazon berendezésekkel ugyanolyan sorrendben kell előállítani.

Többcélú rendszer (multipurpose plant)

Olyan termelő rendszer, ahol a termékek előállítási módja különbözik.

S-gráf (S-graph)

Irányított gráf az ütemezési feladatok leírásához.

Recept-gráf (recipe-graph)

A receptet leíró S-gráf.

Ütemezési-gráf (schedule-graph)

Megoldást leíró S-gráf.

Komponens-gráf (component-graph)

Egy berendezés működését leíró S-gráf.

Betáplálási-sorrend gráf (feed-precedence graph)

Egy taszk bemeneteinek időzítését leíró S-gráf.

Kombinatorikus algoritmusok

A disszertációban bemutatott szétválasztás és korlátozás algoritmus részeként egy körkereső algoritmus (41. ábra) segítségével lehet elvégezni egy részfeladat megvalósíthatósági vizsgálatát és egy leghosszabb út kereső algoritmussal (42. ábra) kaphatunk alsó korlátot a részfeladathoz. Az előzetes körfelismerő algoritmus egy út kereső algoritmust használ (43. ábra). Részletesebb algoritmusok találhatók például Cormen és társai (1997) munkájában.

```
procedure cycle_search(i, LIST)
begin
    if pred(i) = ∅ then
        return no_cycle;
    for all j ∈ pred(i) do
        if j ∈ LIST then
            return cycle;
        LIST = LIST ∪ {j};
        cycle_search(j, LIST);
        if (cycle) then
            return cycle;
        LIST := LIST \ {j};
    return no_cycle;
end

procedure pred(i)
begin
    pred(i) = azon csomópontok halmaza, amelyekből van él i-be;
end
```

41. ábra: Körkereső algoritmus.

```
procedure longest_path (G(N, A))
begin
    for all j ∈ N do
        d(j) := 0;
    for all i ∈ N do
        LIST = {i}; longest = 0;
        while LIST ≠ ∅ do
            LIST = LIST \ {i};
            for each arc (i, j) ∈ A do
                if d(j) < d(i) + c(i, j) then
                    d(j) = d(i) + c(i, j);
                    longest = max(longest, d(j));
                    if j ∉ LIST then
                        tegyük j-t a LIST-be;
end
```

42. ábra: Leghosszabb út kereső algoritmus.

```
procedure path_search (G(N, A), k, l)
begin
    LIST = {k}
    while LIST ≠ ∅ do
        LIST = LIST \ {i};
        for each arc (i, j) ∈ A do
            if j = l then
                return path;
            if j ∉ LIST then
                tegyük j-t a LIST-be;
    return no_path;
end
```

43. ábra: Út kereső algoritmus.

Állítások és bizonyítások

1. Állítás: Egy tetszőleges ütemezési-gráfot ki lehet egészíteni az egy termékhez tartozó batch-ek első csomópontjai között segéd-élekkel, nem okoznak kört és nem növelik meg a leghosszabb út hosszát.

Bizonyítás: Minden ütemezésnél a batch-ek első tevékenységei adott időpontban kezdődnek, mely szerint sorrendbe lehet őket rakni. Tegyük fel, hogy a kezdési idők szerinti nem csökkenő sorrendben húzzuk be a segéd-éleket.

Ha az i-edik batch-hez tartozó első csomópont $(i,1)$ értékei kisebbek, mint az i+1-edik batch-hez tartozóé $(i+1,1)$, tehát nincs út $(i+1,1)$ csomópontból $(i,1)$ csomópontba, így a segéd-élek nem okoznak kört. Ha a két érték egyenlő, akkor sincs út $(i+1,1)$-ből $(i,1)$-be, mert NIS stratégia esetén az első csomópontokból csak recept-él indulhat ki, amelynek súlya nem nulla, azaz nincs olyan nulla hosszú kör, mely tartalmaz első csomópontot. Tehát a segéd-élek nem okoznak kört.

A leghosszabb út definíciója szerint bármely csomópont értéke nagyobb vagy egyenlő, mint egy bemenő élének hossza plusz a bemenő él kiinduló csomópontjának értéke. Ha nulla hosszúságú élet húzunk be egy csomópontból egy nála nem kisebb értékűbe, akkor ez teljesíti a feltételt, tehát nem változtatja meg a leghosszabb út értékét. □

2. Állítás: Ha minden taszkhoz pontosan egy berendezés tartozik, a segéd-éleket a batch-ekhez tartozó összes ekvivalens csomópont között fel lehet venni, az elsővel azonos irányban.

Bizonyítás: Tegyük fel, hogy van olyan megvalósítható megoldás, ahol az *i*-edik taszkhoz és az *i*+1-edik taszkhoz tartozó csomópontok ütemezése különbözik két batch (*j*-edik és *k*-adik) sorrendjében.

- Az SG2 feltételből következően az *i*-edik taszkhoz tartozó berendezés ütemezése miatt létezik olyan út (j,i) és (k,i) csomópontok között, amely tartalmazza a (j,i) és $(j,i+1)$ közötti recept-élet. Továbbá van recept-él (k,i) és $(k,i+1)$ között, tehát létezik út $(j,i+1)$ csomópontból $(k,i+1)$-be.
- Az $i+1$-edik taszkhoz tartozó berendezés ütemezése miatt létezik út $(k,i+1)$ csomópontból, $(j,i+1)$-be.

Mivel létezik út $(j,i+1)$ csomópontból $(k,i+1)$-be, valamint $(k,i+1)$ csomópontból, $(j,i+1)$-be, kör van az S-gráfban, azaz nem megvalósítható megoldást jelöl, ellentmondáshoz jutottunk. Ebből következik, hogy a batch-ekhez tartozó összes tevékenységet ugyanúgy rendezhetjük segéd-élekkel. □

3. Állítás: Ha minden taszkhoz pontosan egy berendezés tartozik, a segéd-él kezdőpontja a recept szerinti következő tevékenységet jelölő csomópontja lehet.

Bizonyítás: Tegyük fel, hogy a *k* berendezést hozzá rendeltük az (i, j) csomóponthoz, ahol az (i, j) csomópont az *i*-edik batch *j*-edik tevékenységét jelöli. Ekkor NIS tárolási stratégia esetén és ha egy taszkhoz csak egy berendezés áll rendelkezésre, a *k* berendezést akkor rendelhetjük hozzá az $(i+1, j)$ csomóponthoz, ha az (i, j) csomópont által jelölt tevékenység befejeződött és az előállított anyag át lett töltve az $(i, j+1)$ csomóponthoz rendelt berendezésbe. Ebből az állítás egyértelműen következik. □

4. Állítás: Ha egy taszk teljesíti a P1 és P2 tulajdonságot, akkor a hozzá tartozó tevékenységek csomópontjai transzformált segéd-élekkel összeláncolhatók a receptben őt megelőzővel azonos irányban.

Bizonyítás: Az előző két állítás bizonyításából következik. □

5. Állítás: Tegyük fel, hogy a *k* berendezés hozzá van rendelve az *i* taszkhoz az aktuális részfeladatban. Ha az *i* taszk elérhető az aktuális részfeladathoz tartozó

részben beütemezett S-gráfban egy irányított úton egy *j* taszkból, akkor nincs olyan megvalósítható megoldás, amelyik az aktuális részfeladaton alapul és ahol a *j* taszk követi az *i* taszkot a *k* berendezés ütemezési sorrendjében.

Bizonyítás: Ha a *k* berendezés ütemezésében a *j* taszk követi az *i* taszkot, akkor létezik út *i*-ből *j*-be. A feltételezés szerint *j*-ből *i*-be is létezik út, tehát kör van az S-gráfban, azaz az ütemezés nem megvalósítható. □