SZAKASZOS FOLYAMATOK ÜTEMEZÉSE AZ S-GRÁF MÓDSZERTAN KITERJESZTÉSEIVEL

DOKTORI (PhD) ÉRTEKEZÉS

Adonyi Róbert
Témavezető: Dr. Friedler Ferenc

Pannon Egyetem
Műszaki Informatikai Kar
Informatikai Tudományok Doktori Iskola

2008

SZAKASZOS FOLYAMATOK ÜTEMEZÉSE AZ S-GRÁF MÓDSZERTAN KITERJESZTÉSEIVEL

Értekezés doktori (PhD) fokozat elnyerése érdekében

Írta: Adonyi Róbert

Készült a Pannon Egyetem Informatikai Tudományok Doktori Iskolája keretében

Témavezető: Dr. Friedler Ferenc

Elfogadásra javaslom (igen / nem)

(aláírás)

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

Veszprém .....

a Szigorlati Bizottság elnöke

Az értekezést bírálóként elfogadásra javaslom:

Bíráló neve: ..... (igen / nem)

(aláírás)

Bíráló neve: ..... (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

Tartalomjegyzék iii

Táblázatok jegyzéke vi

Ábrák jegyzéke viii

Kivonat xi

Abstract xii

Abstrakt xiii

Az értekezésben használt rövidítések xiv

Köszönetnyilvánítás xvii

1. Bevezetés 1
- 1.1. Célkitűzések . . . 2
- 1.2. Jelölések . . . 3
- 1.3. Gyakran használt fogalmak . . . 4

2. Szakirodalom áttekintése 5
- 2.1. Ütemezési feladatok típusai . . . 7
- 2.2. Ütemezési feladatok bonyolultsága . . . 11
- 2.3. Ütemezési feladatok a szakirodalomban . . . 12

3. S-gráf módszertan bemutatása 22
- 3.1. Szakaszos ütemezési feladat megadása . . . 23
- 3.2. Szakaszos folyamatok ábrázolása S-gráffal . . . 26
- 3.3. S-gráf matematikai leírása [86] . . . 28
  - 3.3.1. Recept-gráf . . . 28
  - 3.3.2. Ütemezési-gráf . . . 30
- 3.4. S-gráf alapalgoritmus ütemezési feladatok megoldására . . . 32
  - 3.4.1. Részfeladat ábrázolása és adatstruktúrák inicializálása . . . 32

3.4.2. Szétválasztás eljárás 33
3.4.3. Korlátozás eljárás 36
3.5. Az EQ-SG algoritmus működésének szemléltetése 38
3.6. Az S-gráf alapalgoritmus keresési terének csökkentése: egy termékből több batch előállítása 44
3.6.1. Technikailag ekvivalens ütemezések 46
3.6.2. Recept-gráf kiegészítése segéd-élekkel 47
3.6.3. Redundanciát kizáró feltételek további élesítése 49
3.6.4. Szemléltető feladat a segéd-élek alkalmazására 50

4. Taszk alapú döntési stratégia ütemezési feladatok megoldása során 52
4.1. Berendezés alapú döntési stratégia előnyei és hátrányai 53
4.2. Algoritmus a taszk alapú döntési stratégia alapján 55
4.2.1. Részfeladat ábrázolása és az adatstruktúrák inicializálása . . . 55
4.2.2. Szétválasztás eljárás 56
4.2.3. Korlátozás eljárás 63
4.2.4. TA-SG algoritmus működésének bemutatása 63
4.3. Több batch egyidejű kezelése 70
4.4. Az EQ-SG és a TA-SG algoritmusok viselkedésének az összehasonlítása 73
4.5. Összefoglalás 81

5. Szakaszos működésű termelő folyamat korlátozott tisztítási költségű ütemezése 82
5.1. Berendezések tisztítását figyelembe vevő ütemezési módszerek szakirodalma 83
5.2. A megoldandó festékipari feladat 87
5.3. S-gráf módszertan alkalmazása a festékgyártási feladatra 88
5.3.1. Keresési tér csökkentése 88
5.3.2. Tároló berendezések ütemezése 89
5.3.3. Lineáris programozási modell a részfeladat végrehajtási idejére érvényes alsó korlát meghatározására 90
5.4. Algoritmus a festékgyártási feladatra 90
5.5. Ipari feladat 91
5.6. Összefoglalás 97

6. Ütemezés hőintegrációval 99
6.1. Szakaszos folyamatok hőintegrációjának szakirodalma 100
6.2. Ütemezési és hőintegrációs feladat kapcsolódása 104
6.3. Szakaszos ütemezési feladat hőintegrációja 106
6.4. Kapcsolódó komponens területek 107
6.4.1. Mester feladat: ütemezés 108
6.4.2. Szakaszos folyamatok hőintegrációja 108
6.5. Szakaszos folyamatok hőintegrációjára kidolgozott algoritmus 109

6.5.1. Időintervallumok időben párhuzamos taszkok kezelésére 109
6.5.2. Szétválasztás eljárás 110
6.5.3. Korlátozás eljárás 111
6.6. Példa szakaszos ütemezési feladat hőintegrációjára 116
6.7. Összefoglalás 119

7. Új tudományos eredmények **121**

8. Major results and summary of accomplishments **123**

A. Komponens-gráf **125**

B. Körkereső algoritmus **127**

C. Leghosszabb-út kereső algoritmus **129**

D. Lineáris programozási modell a részfeladat alsó korlátjának számítására **131**

Irodalomjegyzék **135**

Táblázatok jegyzéke

3.1. A 3.1. feladat megoldása során vizsgált részfeladatokra számított alsó korlát értékek (azonosítók a 3.11 ábrához kapcsolódnak). 43
3.2. A 3.2. feladat receptje. 43

4.1. A 4.2. feladat receptje. 66
4.2. A 4.10 ábra keresési fájának csúcsaihoz tartozó ütemezések. 68
4.3. A 4.3. feladat receptje. 70
4.4. A 4.4. feladat receptje. 72
4.5. A 4.4. feladat megoldása során a TA-SG algoritmussal kapott futási eredmények. 74
4.6. A 4.5. feladat receptje. 75
4.7. A 4.5. feladat megoldása során kapott futási eredmények. 75
4.8. A 4.6. feladat receptje. 77
4.9. A 4.7. feladat receptje. 78

5.1. Ipari feladat receptje. 93
5.2. Batch-ek száma termékenként. 94
5.3. Örlő berendezések ($E1$-$E5$ berendezések) tisztítási költségei (CU). . . 94
5.4. Keverő berendezések ($E6$-$E9$ berendezések) tisztítási költségei (CU). . 96
5.5. Tároló berendezések ($E10$-$E20$ berendezések) tisztítási költségei (CU). 96

6.1. Szemléltető példa receptje. 104
6.2. A hőintegrációs feladat receptje. 117
6.3. A hőintegrációs feladat hőáramai. 117
6.4. A feladat optimális megoldásában levő hőcserék. 119

D.1. A szemléltető példa receptje. 132

Ábrák jegyzéke

3.1. Ütemezés ábrázolása Gantt-diagrammal. 25
3.2. Téglagyártás receptje. 26
3.3. Hagyományos gráf átalakítása kombinatorikus algoritmusok számára. 27
3.4. Élek jelentése az S-gráfban. 28
3.5. Az EQ-SG algoritmus. 34
3.6. Az EQ-SG algoritmus *EQ-szétválasztás* eljárása. 37
3.7. Az EQ-SG algoritmus *EQ-korlátozás* eljárása. 38
3.8. A 3.1. feladat recept-gráfja. 39
3.9. A 3.1. feladat teljes leszámolási fája (korlátozási lépés nélkül). 40
3.10. A 3.1. feladat teljes leszámolási fájának leveleihez tartozó S-gráfok. Körök vastagított élekkel jelölve. 41
3.11. A 3.1. feladat megoldása során bejárt keresési fa a korlátozás lépés alapján. (Az optimális megoldás a 11-es részfeladat, a végrehajtási ideje: 43.) 42
3.12. A 3.2. feladat recept-gráfja. 44
3.13. A 3.2. feladat egy optimális ütemezési-gráfja. 45
3.14. A 3.2. feladat 3.13. ábrán látható ütemezésének Gantt-diagrammja. . 45
3.15. Ütemezés Gantt-diagrammja. 46
3.16. A 3.15 ábra ütemezésével technikailag azonos ütemezés. 46
3.17. A 3.16 ábra ütemezését kizáró recept-gráf. 47
3.18. Egyszerű recept recept-gráfja segéd-élekkel kiegészítve, ha n batchnyit kell egy termékből ütemezni. 48
3.19. Összetett recept recept-gráfja segéd-élekkel kiegészítve. 49

3.20. Recept-gráf segéd-élekkel kiegészítve, ha minden taszk csak egy berendezéssel hajtható végre. 50

4.1. A *TA-SG* eljárás. 57
4.2. A taszk alapú döntési stratégiát megvalósító *TA-szétválasztás* eljárás a TA-SG algoritmus számára. 59
4.3. A szülő részfeladat ütemezése (leghosszabb út: 47). 60
4.4. A gyermek részfeladat ütemezése (leghosszabb út: 45). 60
4.5. A *TA-szétválasztás* eljárás során új taszk ütemezése a gyermek részfeladatban. 62
4.6. 4.1. feladat feladat recept-gráfja. 64
4.7. A 4.1. feladat EQ-SG és TA-SG algoritmusokkal bejárt keresési fái. . 65
4.8. A 4.1. feladat optimális ütemezését ábrázoló ütemezési-gráf és Gantt-diagramm. 66
4.9. A 4.2. feladat recept-gráfja. 66
4.10. A 4.2. feladat teljes leszámolási keresési fája. 67
4.11. A TA-SG algoritmus során bejárt keresési fa. 69
4.12. A 4.2. feladat egy optimális megoldásának ütemezési-gráfja. 69
4.13. A 4.2. feladat egy optimális megoldásának Gantt-diagrammja. 70
4.14. A 4.3. feladat recept-gráfja. 71
4.15. A 4.3. feladat egy optimális ütemezési-gráfja. 71
4.16. A 4.3. feladat egy optimális ütemezésének Gantt-diagrammja. 71
4.17. A 4.4. feladat recept-gráfja. 73
4.18. A 4.5. feladat recept-gráfja (1-1 batchnyi minden termékből). 75
4.19. A 4.6. feladat recept-gráfja. 76
4.20. A 4.6. feladat egy optimális ütemezésének Gantt-diagrammja. 77
4.21. A 4.7. feladat recept-gráfja. 79
4.22. A 4.7. feladat egy optimális ütemezésének Gantt-diagrammja. 79

5.1. Festékgyártást leíró recept. 87
5.2. Recept-gráf készítése a festékgyártás receptjéből. 89

5.3. A 9-es és 7-es csúcs közötti él biztosítja, hogy az $E7$ berendezés, csak a 4-es taszk (csomagolás típusú) befejezése után kezdheti meg a 7-es taszkot. 90
5.4. A *P-korlátozás* eljárás. 92
5.5. Minimális végrehajtási idejű ütemezési-gráf, melyben a szaggatott élek tisztítási költséggel járó váltásokat jelölnek. 95

6.1. A 6.1. táblázat recept-gráfja. 105
6.2. Minimális végrehajtási idejű ütemezés Gantt-diagrammja. 105
6.3. Egy megvalósítható ütemezés Gantt-diagrammja. 106
6.4. Recept-gráf kiegészítve hőfolyamatokkal. 108
6.5. S-gráf időintervallumokkal. 110
6.6. Az *SCH-HENS-korlátozás* eljárás. 111
6.7. Párhuzamos meleg- és hidegáramok (a vonalazott terület a hőcsere lehetséges idejét jelöli). 113
6.8. S-gráf lehetséges hőcserével. 115
6.9. Az i taszk tevékenységeinek sorozata hőcsere esetén. 115
6.10. A 6.1. feladat recept-gráfja. 118
6.11. A feladat optimális ütemezésének Gantt-diagrammja. 119

A.1. Ütemezési-gráf a komponens-gráfok bemutatásához. 125
A.2. A G'_{E1}, G'_{E2}, G'_{E3} és G'_{E4} komponens-gráfok. 126

B.1. A körkereső algoritmus. 128

C.1. A leghosszabb út kereső algoritmus. 130

D.1. A szemléltető példa recept-gráfja. 133

Kivonat

Szakaszos folyamatok ütemezése az S-gráf módszertan kiterjesztéseivel

Termelő és szolgáltató rendszerek működését, viselkedését elemezve az esetek jelentős részében ütemezési feladatokat kell megoldani. Az ütemezési feladatok kombinatorikus jellege és az ipari feladatok mérete miatt a szakirodalomban közölt matematikai programozási modellek nagy számú egész típusú változót tartalmazhatnak. A nagy méretű modellek megoldásának nehézsége miatt gyakran heurisztikus szabályokkal csökkentik a keresési teret, ami az optimum elvesztéséhez vezethet.

A dolgozat az ütemezési feladatok matematikai modelljének megoldására az S-gráf módszertant [Holczinger 2002, Sanmartí 1998, 2002] használja. Az S-gráf módszertan kihasználja az ütemezési feladatok speciális tulajdonságait, a keresési tér csökkentésére, ezáltal nagyméretű ipari feladatok megoldását teszi lehetővé. Az S-gráf módszertan tartalmazza az S-gráfot, ami megfelelően ábrázolja az ütemezési feladatokat, és az alapalgoritmust, ami kombinatorikus eszközökkel kiegészítve ipari méretű feladatok megoldását teszi lehetővé.

A disszertációban a szerző különböző ütemezési feldatokat old meg az S-gráf módszertannal. Először egy egy új szétválasztási algoritmust ismertet, mellyel eddig kezelhetetlen méretű feladatok megoldására nyílik lehetőség. Másodszor a szerző bemutatja az S-gráf módszertant kiterjesztését a berendezések ütemezésétől függő tisztítási költségének kezelésére. Végezetül a szerző egy S-gráf módszertant használó algoritmust dolgozott ki, mellyel ütemezési és hőintegrációs feladatok egyidejű megoldására nyílik lehetőség.

Abstract

Batch process scheduling with the extensions of the S-graph framework

The analysis of production and supply systems usually requires the solution of a scheduling problem. Because of their combinatorial nature and the significant size, industrial scheduling problems in the literature are described by mathematical models containing large number of integer variables. The computational difficulties in solving such large-scale mathematical models, usually call for the reduction of the search space using heuristic rules, which can lead to the loss of optimality.

In the present dissertation the S-graph framework [Holczinger 2002, Sanmartí 1998, 2002] is used for solving scheduling problems. This framework exploits the special features of scheduling problems to efficiently reduce the search space, thus enabling the solution of realistic industrial-scale cases. The framework consists of the S-graph, which provides an appropriate representation of scheduling problems and an algorithm employing combinatorial tools for solving industrial-size scheduling problems.

Several types of scheduling problems have been solved using the S-graph framework. Firstly, a new branching procedure is introduced, which achieves further acceleration to solve certain scheduling problems. Secondly, the S-graph framework is extended to solve the scheduling problems, taking into account cleaning costs in industries where it is relevant. Finally, an algorithm which is capable of solving combined scheduling and heat integration problems is introduced, based on the S-graph framework.

Abstrakt

Die Beschreitungen der S-Graf Methode mit den Taktaufgaben von Batch-Prozessen

Bei der Analyse vom Betrieb und Wesen der Produktions- und Leistungssysteme müssen meistens Taktaufgaben gelöst werden. Da die Taktaufgaben einen kombinatorischen Charakter haben und da bei der Produktionsindustrie wachsende Ausmasse annehmen, in der Fachliteratur angegebene mathematische Programiermodelle können viele integer Variablen beinhalten. Oft wird der gesuchte Suchraum wegen der schwierigen Lösungen von grossen Modellen mit heuristischen Regeln verringert. Es führt zum Verlust vom Optimum.

Die Arbeit verwendet die S-Graf Methode [Holczinger 2002, Sanmartí 1998, 2002] zur Lösung der mathematischen Modell von Taktaufgaben. Die S-graf Methode nutzt die spezifischen Eigenschaften der Taktaufgaben aus, um den Suchraum zu verringern und so können bedeutende industrielle Aufgaben gelöst werden. Die S-Graf Methode enthält den S-Graf - der stellt die Taktaufgaben entsprechend dar - und den Grundalgorithmus, der ergänzt von kombinatorischen Mitteln geeignet ist, bedeutende industrielle Aufgaben zu lösen.

In dieser Abhandlung werden mit der Hilfe von S-Graf Methode verschiedene Taktaufgaben verwirklicht. Zuerst legt der Verfasser ein neuer Bound-Algorithmus dar, damit bisher unlösbare Grösse von Aufgaben gelöst werden können. Dann veranschaulicht der Verfasser die Beschreitung von S-Graf zur Abfertigung der Reinigungskosten anhand der Taktaufgaben von Einrichtungen. Am Ende der Arbeit konzipierte der Verfasser eine S-Graf Methode bei einem Algorithmus, womit Taktaufgaben und Wärmeintegrationsaufgaben gleichzeitig gelöst werden können.

Az értekezésben használt rövidítések

ACS (Ant Colony Systems) – A hangyák viselkedését modellező evolúciós módszer optimalizálási feladatok megoldására.

AoA (Activity on Arc) – Olyan gráf, melyben az ütemezési feladat tevékenységeit a gráf élei jelölik.

AoN (Activity on Node) – Olyan gráf, melyben az ütemezési feladat tevékenységeit a gráf csomópontjai jelölik.

CIS (Common Intermediate Storage) – Tárolási stratágia, melyben adott mennyiségű tároló berendezést használhatunk, azonban a tároló berendezések helye nem rögzített a receptben.

CPU (Central Processing Unit) – Központi feldolgozó egység, vagy processzor a számítógép azon része, mely az utasítások értelmezését és végrehajtását vezérli.

EQ-SG – Az S-gráf módszertanhoz kifejlesztett berendezés alapú döntéseket használó szétválasztás és korlátozás algoritmus.

FIS (Finite Intermediate Storage) – Olyan tárolási stratégia a termelési folyamatban, ahol véges számú és méretű tároló berendezés ütemezésével kell gondoskodni a köztes anyagok tárolásáról.

FMS (Flexible Manufactoring System) – Olyan termelő rendszer, ahol a taszkok végrehajtási ideje változhat.

HEN (Heat Exchanger Network) – Hőcserélő berendezéseket tartalmazó hálózat.

HENS (Heat Exchanger Network Synthesis) – Hőcserélő hálózatok szintézise egy olyan feladat, ahol a hőcserélő hálózat megtervezése a cél.

ISA SP88 – Szakaszos folyamatok leírására létrejött ISA szabvány.

JIT (Just In Time) – Az ütemezés célja, hogy a termékek minél pontosabban a kívánt határidőre készüljenek el.

LCA (Lifecycle Analysis) – Az LCA során az optimális gyártási struktura kialakításában nemcsak a termékek előállításának költségeit, hanem a lebontási, újrahasznosítási költségeit is figyelembe veszik.

LP (Linear Programming) – Lineáris programozási feladat.

MIBLP (Mixed Integer Bilinear Programming) – Olyan optimalizálási feladat, ami szétbontható két összefüggő MILP feladatra.

MILP (Mixed-Integer Linear Programming) – Optimalizálási feladatot leíró egész és folytonos változókat és lineáris feltételeket tartalmazó matematikai programozási modell (vegyes egész lineáris programozási feladat).

MINLP (Mixed-Integer NonLinear Programming) – Egy optimalizálási feladatot leíró egész és folytonos változókat és nemlineáris feltételeket tartalmazó matematikai programozási feladat (vegyes egész nemlineáris programozási feladat).

MIS (Mixed Intermediate Storage) – Olyan tárolási stratégia, melyben a gyártási folyamat különböző pontjain UIS, NIS, FIS és ZW is jelen lehet.

NIS (Non Intermediate Strorage) – Olyan tárolási stratégia a termelési folyamatban, ahol a köztes anyagok berendezésekben való tárolásáról gondoskodni kell az ütemezés során.

P-gráf – Hálózatszintézis feladatok ábrázolására és megoldására létrehozott páros gráf. A P betű az angol process szóra utal.

PERT (Program Evaluation and Review Technique) – Projekt menedzsment során tevékenység ütemezésre és elemzésére használt eszköz.

RTN (Resource Task Network) – Ütemezési feladatok ábrázolására létrehozott páros gráf.

S-gráf – Ütemezési feladatok ábrázolására létrehozott gráf. Az S betű az angol schedule szóra utal.

SCM (Supply Chain Management) – Ellátási lánc menedzsment a termékek előállításával és értékesítésével kapcsolatban jelentező anyag- és információáramokkal, pénzügyi folyamatok modellezésével és optimalizálásával foglalkozó tudományterület.

SS/TDMA (Satellite-Switched/Time Division Multiple Access) – Műhold - földi hálózat csatolás ütemezési feladat.

SSN (State Sequence Network) – Ütemezési feladatokhoz létrehozott gráf, melyek az állapotok (state) sorrendjét ábrázolja.

STN (State Task Network) – Ütemezési feladatok ábrázolására létrehozott páros gráf.

TA-SG – Az S-gráf módszertanhoz kifejlesztett taszk alapú döntéseket használó szétválasztás és korlátozás algoritmus.

UIS (Unlimited Intermediate Storage) – Olyan termelési folyamat, ahol az anyagok, vagy a recept jellege miatt a köztes anyagok tárolásáról nem kell gondoskodni az ütemezés során.

ZW (Zero Wait) – Olyan tárolási stratégia, ahol a köztes anyagokat nem lehet tárolni, hanem előállításuk után azonnal fel kell dolgozni őket.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Friedler Ferenc professzor úrnak, folyamatos útmutatásáért és támogatásáért, mellyel a bemutatásra kerülő eredményeim és PhD dolgozatom megszületését segítette. Köszönöm minden kollegámnak a kreatív, együttműködő és jó hangulatú légkört amiben dolgozhattam. Mindezek felett szeretném megköszönni családomnak azt a céltudatos, elszánt és kitartó ösztönzést és támogatást, mellyel tanulmányaim során elkísértek.

1. fejezet

Bevezetés

Termelő és szolgáltató rendszerek működését, viselkedését elemezve az esetek jelentős részében ütemezési feladatokkal találkozunk, így például a vegyiparban, az olajiparban, a gépiparban, a mezőgazdaságban, az építőiparban és a szállítmányozásban. A termelő és szolgáltató rendszerek mellett az informatikai rendszerekhez is kapcsolódnak bonyolult ütemezési feladatok. Ha a rendszer tartalmaz olyan konkurens folyamatokat, melyek ugyanazokat az erőforrásokat igénylik és nem áll rendelkezésre megfelelő számú erőforrás, akkor az erőforrások ütemezésével lehet a folyamatokat kiszolgálni.

Az ütemezés a számítástechnikának is az egyik kulcskérdése. A számítógép processzorának (CPU) működését ütemezési algoritmusok irányítják. A végrehajtásra váró folyamatok processzorra való ütemezésének jósága, hatékonysága a számítógép viselkedését, használhatóságát befolyásolja. Mivel az ütemezés minősége az egész rendszer működését meghatározza, ezért nagyon fontos, hogy a nagy számú lehetséges ütemezési alternatíva közül a lehető legjobb ütemezéseket keressük meg elfogadható számítási idő alatt.

Az Internetet működtető eszközök között is találunk olyan összetett rendszereket, melyek megfelelő működéséért ütemezési algoritmusok a felelősek. Képzeljünk el egy kliens-szerver architektúrán létrehozott on-line digitális könyvtárat, mely több nagy teljesítményű szerver gépből áll. Ezeknek a szervereknek kell a kliens gépeket adatokkal kiszolgálniuk oly módon, hogy a rendszer válaszideje a lehető legkisebb legyen.

Ilyenkor a kliens kérések megfelelő szerver gépekhez rendelését ütemezni kell.

Ütemezés során a cél erőforrásainkhoz taszkok hozzárendelése egy olyan idő intervallumra mely alatt a taszkot elvégzi. Azt az erőforrás – taszk hozzárendelést keressük, mely optimális (pl. minimális végrehajtási idejű, legnagyobb profitú) és teljesíti a rendszer korlátozásait (pl. taszkok sorrendisége, erőforrások tisztítása).

1.1. Célkitűzések

Az értekezésben célom szakaszos műveleteket tartalmazó ütemezési feladatok bemutatása, optimális megoldásukra algoritmikus eszközök adása. Munkámhoz a nagy méretű ütemezési feladatokat is hatékonyan kezelő Sanmartí és társai által publikált S-gráf módszertant [84, 86] tekintem alapnak az értekezésben érintett feladatosztályok vizsgálatánál. Az S-gráf módszertan hatékonysága az ütemezési feladatok kombinatorikus tulajdonságainak kihasználásán alapszik [34].

Az S-gráf módszertan tartalmazza az ütemezési feladatok megfelelő ábrázolására kidolgozott S-gráfot, az alapalgoritmust, mely az S-gráf ábrázolást használva megadja a feladat optimális megoldását és a lehetőséget arra, hogy kombinatorikus eszközök beépítésével további feladatok megoldására nyílik lehetőség. Az értekezésben bemutatok egy új szétválasztás eljárást, mely eljárás az S-gráf alapalgoritmus döntési stratégiájára egy másik alternatíva. Az új szétválasztás eljárást az S-gráf módszertanba építem. Bemutatom az új eljárás működését és ismertetem kedvező tulajdonságait (4. fejezet).

Az S-gráf módszertanban a berendezések optimális ütemezéséhez figyelembe veszszük a berendezések váltási idejét, azaz a berendezéshez rendelt két egymásután végrehajtandó taszk között a berendezés beállítására, konfigurálására szükséges időmennyiséget. Az S-gráf módszertan azonban nem számol a berendezések tisztításának szükségességével, a tisztítások költségeivel. Bemutatok egy algoritmust, mely olyan optimális ütemezést szolgáltat, melyben a berendezések tisztításának költségeit is figyelembe veszem (5. fejezet).

Szakaszos folyamatokat tartalmazó ütemezési feladatok gyakran olyan anyagáramokat is tartalmaznak, melyeket hűteni, vagy melegíteni kell. Az anyagáramok hűtése és melegítése hőcsere útján történik úgy, hogy a hőcseréknél a rendszer más anyagáramait, vagy külső szolgáltatótól vásárolt hőt használhatunk. Megfelelő ütemezéssel a külső szolgáltatótól vásárolt hő mennyisége csökkenthető. Bemutatok egy algoritmust, melyben a berendezések ütemezése mellett figyelembe veszem a termelő rendszer hűtési és melegítési igényeit, megadom a hőintegrációs ütemezési feladatok optimális megoldását (6. fejezet).

1.2. Jelölések

A termelő folyamatban ütemezendő termékeket az A, B, C, ... jelöli. Az ütemezési feladatban a taszkokat pozitív egész számokkal $(1, 2, 3, ...)$, vagy i, j, k változókkal jelölöm. Az ütemezendő n darab berendezést $E1$, $E2$, ..., En jelöli.

Az S-gráf egy egy olyan diszjunktív gráf, melyben nemnegatív értékű élek szerepelnek. Ha a gráfban valamely élnek nincs feltüntetve az értéke, akkor az az él 0 értékű élt jelöl. Az S-gráfban szereplő csúcsokat az egyértelmű hivatkozás céljából egy pozitív egész azonosítóval látom el. Ütemezési feladatban ha valamely berendezéshez nincsen váltási idő definiálva, akkor a váltási idő értéke 0.

A szétválasztás és korlátozás algoritmusok működését keresési fákkal szemléltetem. A keresési fa csúcsai a részfeladatok, az élei részfeladatokon végrehajtott döntéseket ábrázolják. A keresési fa csúcspontjaiban a vizsgált döntési változót az élein a vizsgált döntési változó lehetséges értékeit ábrázolom.

Az ütemezési szakirodalom nyelve az angol. A gyakran használt fogalmak, elnevezések, rövidítések angol nyelvű megfelelőjét a magyar kifejezés után zárójelben dőlt betűkkel adom meg.

1.3. Gyakran használt fogalmak

Ütemezési feladatokban a feladat jellegétől függően eltérő elnevezést használnak azonos, vagy hasonló jelentésű fogalmakra, eszközökre. Megadom az értekezésben használt elnevezéseket és a teljesség igénye nélkül ismertetek néhány alternatívát az elnevezésekre.

- Termék (*product*): az ütemezés során előállítandó anyag, tárgy, szolgáltatás.
- Recept (*recipe*): a termék előállítását leíró gyártási utasítások.
- Batch (*batch*): a termék előállításának egyszeri folyamata, mely során adott mennyiségű termék keletkezik. A shop ütemezési feladatoknál a munka (*job*) elnevezést használják a szakirodalomban. A batch magyar nyelven köteget, adagot jelent, azonban a vegyiparban a francia eredetű sarzs (charge) szót használják. Az értekezésben a megfelelő magyar kifejezés hiánya miatt a továbbiakban batch-et használom.
- Taszk (*task*): a termék előállításának egy elemi lépése, mely során adott bemenetből adott kimenet keletkezik. A taszkokat a szakirodalomban szokás műveletnek, munkának is nevezni.
- Berendezés (*equipment unit*): a taszk végrehajtására használható eszköz. Szokás még gépnek is nevezni.

2. fejezet

Szakirodalom áttekintése

A folytonos, félfolytonos és a szakaszos műveleteket tartalmazó termelő, szolgáltató rendszerek ütemezése fontos feladat. Az elméletben és a gyakorlatban is nagy figyelmet szentelnek az ütemezési feladatok megoldására a mai napig. Az ütemezés jelentősen befolyásolja a termelés hatékonyságát, ezért gazdaságossági szempont a megfelelő termelési folyamat, ütemezés meghatározása. A termelési rendszer belső rugalmassága lehetőséget adhat egy jobb ütemezés megvalósítására. Az egyre pontosabb, összetettebb ütemezési modellek és a folyamatosan növekvő számítási teljesítmény indukálja az új tudományos eredményeket és biztosítják az ipari alkalmazhatóságukat.

Vegyipari termékek jelentős részét szakaszos gyártási műveletekkel állítják elő. Szakaszos gyártási folyamatok legjelentősebb tulajdonsága a nagy mértékű rugalmasság, mellyel nagyszámú, különböző terméket lehet előállítani. A szakaszos folyamatok széles körben való alkalmazása miatt fontos az ilyen termelő rendszerek ütemezésének kutatása.

Szakaszos rendszerekhez kapcsolódó ütemezési feladatokat csoportosíthatjuk a műveletek és berendezések összekapcsolási szabályai alapján. Többtermékesnek nevezzük azt a rendszert (*multiproduct plant*), melyben az egy termékhez tartozó több batchnyi mennyiséget pontosan ugyanazokkal a berendezésekkel és ugyanabban a sorrendben lehet előállítani. Többcélú rendszer (*multipurpose plant*) esetén a termékek előállítási módja különböznek.

Szakaszos termelő rendszerekben a termelés batchekben történik. Ez azt jelenti, hogyha az előállítandó termék mennyisége több, mint amennyi termék a recept egyszeri végrehajtásával keletkezik, akkor a recept többszöri megismétlésével állítják elő a kívánt termékmennyiséget. A recept egyszeri végrehajtását, ütemezését jelenti egy batchnyi termék előállítása.

Termelő folyamatoknál fontos kérdés, hogy a gyártás során keletkező köztes anyagok milyen tulajdonságúak. Tárolás szempontjából kérdéses, hogy a köztes anyagokat lehet-e tárolni, kell-e tárolni, vagy a folyamat és a gyártási környezet olyan, hogy a köztes anyagok tárolási kérdéseivel nem kell foglalkozni az ütemezés során. Ha tárolni kell a köztes anyagokat, akkor azokat csak a tárolásra használt dedikált tároló berendezésekben, vagy magában az anyagot gyártó berendezésben lehet tárolni addig, míg a következő gyártó berendezésbe nem kerül. Az ütemezési feladatokban előforduló tárolási tulajdonságok a következő fő tárolási stratégiákkal jellemezhetőek [33, 80]:

- Az *UIS* (*Unlimited Intermediate Storage*) tárolási stratégia esetén a gyártási környezet olyan, hogy a gyártási folyamat alatt keletkező köztes anyagokat végtelen mennyiségben lehet tárolni. Ez jelentheti azt, hogy megfelelően nagy számban áll a rendelkezésünkre tároló berendezés, és ezeknek ütemezésével nem kell foglalkozni, vagy esetleg azt is, hogy a köztes anyag olyan tulajdonságú, hogy nem szükséges tárolót biztosítani a számára.

- Az *NIS* (*Non Intermediate Storage*) tárolási stratégia esetén taszkok közötti köztes anyag tárolására nincsen lehetőség, azaz a köztes anyagot a taszk elvégzése után a taszkot végrehajtó berendezésben kell tárolni addig, míg az ütemezés alapján a következő taszkot végrehajtó berendezésbe nem kerül a köztes anyag. Ennek a korlátnak az a következménye, hogy a taszkot végrehajtó berendezés csak azután hajthatja végre a hozzá rendelt következő taszkot, ha a benne tárolt köztes anyagot áttöltöttük egy másik berendezésbe.

- Az *FIS* (*Finite Intermediate Storage*) tárolási stratégia esetén a termelő folyamat véges számú és méretű tároló berendezést tartalmaz. Mindegyik tároló

berendezés csak két előre meghatározott, rögzített berendezés között alkalmazható a köztes anyag tárolására. Általában a tároló berendezések megfelelő ütemezését is meg kell határozni.

- A *ZW* (*Zero Wait*) tárolási stratégia esetén a köztes anyagokat nem tárolhatjuk sem az azokat előállító berendezésekben, sem dedikált tároló berendezésekben, hanem a köztes anyagokat az előállításuk után azonnal át kell tölteni az azt feldolgozó következő berendezésbe, amely az áttöltés után rögtön elkezdi feldolgozni a benne levő anyagot. Szemléletesen az ütemezésnek biztosítania kell ZW stratégia esetén, hogy a köztes anyagok sehol se várakozhatnak a termelő rendszerben.

- Az *MIS* (*Mixed Intermediate Storage*) tárolási stratégia az előző négy tárolási stratégia keveréke, azaz a gyártási folyamat különböző pontjain más-más tárolási stratégiák teljesülését kell biztosítani.

- A *CIS* (*Common Intermediate Storage*) tárolási stratégia esetén a termelő rendszer véges számú és méretű tárolót tartalmaz. Ez a stratégia abban különbözik az FIS stratégiától, hogy ebben az esetben a tároló berendezések helye nem rögzített.

A szakirodalomban elsősorban az UIS stratégiával foglalkoznak. Az UIS stratégia elsősorban a gépiparra jellemző, ahol a köztes anyagok tárolása könnyen megoldható egy nagy raktárépület segítségével. Az NIS stratégia a vegyipari rendszerekre jellemző, mikor folyékony vagy akár instabil köztes anyagok is szerepelhetnek a gyártási folyamatban. Az ilyen anyagok megfelelő tárolásáról a berendezések ütemezésének kell gondoskodnia.

2.1. Ütemezési feladatok típusai

Ütemezési feladatokat széles körű előfordulásuk és összetett jellegük miatt sokféle szempont alapján osztályozhatjuk. Az ütemezési feladatokat osztályozhatjuk a termékek elkészítéséhez szükséges műveletek, feladatok végrehajtásának módja alapján.

Megkülönböztethetünk „job shop", „flow shop" és „open shop" ütemezési feladatokat. Minden termék egy munka (*job*) meghatározott szabályok szerinti végrehajtásával állítható elő. A job egy, vagy több műveletet tartalmaz, a műveleteket berendezések hajtják végre. Általában feltehetjük, hogy az elkezdett műveleteket nem lehet megszakítani, minden műveletet egy időben csak egy gépen lehet végrehajtani.

A „flow shop" ütemezési feladatokban a job-ok műveletei ugyanabban a sorrendben haladnak végig az azokat végrehajtó ugyanazon berendezéseken. Ezért a flow shop ütemezést „permutációs ütemezésnek" szokás nevezni, mivel a feladat a termékek optimális sorrendjének, permutációjának a meghatározása.

A „job shop" ütemezési feladatokban mindegyik job több gépen keresztül, azonban a műveletek rögzített sorrendjében végezhető el. Flow shop feladatokkal ellentétben job shop ütemezés esetén a különböző jobok azonos műveletei különböző gépekkel is elvégezhetőek. Feltételezzük, hogy különböző job-ok műveletei között nincsen rendezés, bármelyiket végre lehet hajtani hamarabb. Blazewicz és társai összegyűjtötték és elemezték a job shop feladatok megoldására született módszereket [11].

Az „open shop" ütemezési feladat hasonló a job shop feladathoz, eltérés abban van, hogy míg a job shop feladatoknál a job-ok műveleteinek a sorrendje rögzített, addig az open shop feladatokban a job műveleteit bármilyen sorrendben végre lehet hajtani a megfelelő gépeken [45].

A flow-, job- és open shop feladatok vizsgálatával jelentős mennyiségű szakirodalom foglalkozik, különböző módszereket dolgoztak ki megoldásukra (pl: heurisztika, szétválasztás és korlátozás, dinamikus programozás, egész programozás, tabu keresés, szimulált hűtés, neurális hálózatok, hangya algoritmusok). A későbbiekben szakirodalmi példákat mutatok be az ütemezésre kidolgozott módszerekre. Guo és társai a több gépes job shop ütemezési feladatok megoldására a egy genetikus algoritmust fejlesztettek ki [29]. Az algoritmus működését ipari feladat megoldásával szemléltették. A job shop feladatot JIT (*Just In Time*) környezetben vizsgálva oldották meg, ahol a rendelések és termékek jellegéből adódóan a termékeket a határidőhöz minél közelebb kell legyártani. A JIT feladatoknál a termék határidőhöz képesti koraiságát is büntetik, hiszen ilyenkor a terméket tárolni kell és ez költséggel jár, illetve a határidőhöz képesti késést is büntetni kell, mert ilyenkor a megrendelő elégedettsége csökken, ami

a gyártó piaci helyzetének romlásához vezethet.

A szakirodalomban az ütemezés során figyelembe vett időintervallum (*time horizon*) alapján többféle osztályozását találhatjuk az ütemezési feladatoknak. Glismann és Gruhn három osztályba sorolta az ütemezési feladatokat [24]. A leghosszabb időintervallumot az üzleti szintű hosszú távú tervezés (*long term planning*) használja, ezt követi a gyártás rövid távú ütemezése (*short term scheduling*), a legrövidebb időintervallum a gyártás irányítása (*controlling*). A hosszú távú tervezés során a vizsgált időintervallum hónapnyi nagyságú, az optimalizálás célja a profit növelése. Általában a feladat a termékekből gyártandó mennyiségek, a rendszer működési paramétereinek a meghatározása. A hosszú távú tervezés eredményét a rövid távú ütemezésnél használjuk fel, ahol a vizsgált időintervallum általában egy-két hét nagyságúra zsugorodik. Ezen a szinten a cél, hogy meghatározzunk egy olyan ütemezést, mely teljesíti a termékek határideit, a hosszú távú tervezésben meghatározott célokat. A rövid távú ütemezés eredményét (pl. egy Gantt-diagramm formájában) használja a gyártás irányítás. Az ütemezés alapján megadhatjuk a gyártás végrehajtásához szükséges folyamat irányítási utasításokat. Bistline és társai még finomabb osztályozást adtak az ütemezési feladatokra [10]. Az ütemezési feladatokat az intervallum hossza alapján öt osztályba sorolták. A leghosszabb időintervallumot a hosszú idejű tervezési feladatok (*long range planning*) fogják át. Ezekben a feladatokban a hosszútávú tervezési kérdések tartoznak. A tervezési időintervallum körülbelül kettő vagy több évben mérhető. A középtávú tervezési feladatokhoz (*medium range planning*) a logisztikai feladatok tartoznak. Az ilyen feladatok egy, vagy két évnyi időtartamúak. A rövid idejű tervezési feladatok (*short range planning*) esetén a termelés szükségleteit vizsgálják. A feladatok három-hat hónap időintervallumúak. Az ütemezési feladatokban a berendezésekhez a taszkok hozzárendelése és ütemezése történik. A feladatok kettő-hat hét időintervallumot fognak át. A legrövidebb időintervallummal a reaktív ütemezési (*reactive scheduling*) feladatok rendelkeznek. Itt követelmény az azonnali ütemezési válasz adása, hogy szükség esetén beavatkozhassunk a gyártásba.

Az ütemezési feladatok sokszínűsége és összetettsége miatt sokfajta szempontot figyelembe vehetünk a feladatok megfogalmazásakor. Méndez és társai összegyűjtötték a szakirodalomban figyelembe vett ütemezéshez kapcsolódó különböző aspektusokat

[53]. A következő tulajdonságokat vizsgálták az ütemezési feladatoknál.

1. A gyártás topológiája alapján megkülönböztethetünk szekvenciális, vagy tetszőleges hálózatokat. A szekvenciális topológia során a berendezések használatának módja alapján job shop és flow shop feladatokkal találkozhatunk.
2. A berendezések taszkhoz hozzárendelési módjai (rögzített, változó) és a berendezések egymás után használhatósága (korlátozott, teljes) befolyásolja az ütemezést.
3. Köztes anyag tárolásának módja alapján.
4. Köztes anyag továbbításának módja (azonnali, időigényes) alapján.
5. Batch mérete (rögzített, változó) alapján.
6. Taszk végrehajtási idő (rögzített, berendezés függő, batch méretétől függő) alapján.
7. Igény jellege (határidő, ütemezési horizont) alapján.
8. Váltási idő (sorrendfüggő, sorrendtől nem függő, berendezés függő) alapján.
9. Erőforrás és idő korlátok (műszakok, javítási idő) alapján.
10. Költség (berendezés, tárolás, váltás) alapján.
11. Determinisztikus, vagy sztochasztikus ütemezési feladat.

Liaw munkájában hibrid genetikus algoritmust használ open shop ütemezési feladatok megoldására [45]. A hibrid módszerben a genetikus algoritmusba integrálja a lokális optimum megtalálására használt tabu kereső algoritmust. Ezzel az ötlettel a genetikus algoritmus keresési tere lecsökken a lokális optimumokat tartalmazó altérre. A kísérletek alapján a hibrid algoritmus az esetek nagy részében megtalálja a feladatok optimális megoldását.

2.2. Ütemezési feladatok bonyolultsága

Az ütemezési feladatokról már korábban bebizonyosodott, hogy algoritmuselméleti szempontból a nehezen megoldható feladatok közé (az ún. NP-teljes) tartozik [22]. Ezért különösen fontos a célszerű modellezési technika és megoldó módszer kidolgozása. Lenstra és Rinnoy Kan bizonyította, hogy a hagyományos job shop feladat az NP teljes feladatok osztályába tartozik [44]. Tzafestas bebizonyította, hogy a rugalmas termelő rendszerek (*Flexible Manufactoring System*, FMS) ütemezése az NP teljes feladatok osztályába tartozik [97]. Ha az open shop feladat két gépes, akkor létezik polinomiális algoritmus az optimális megoldás megkeresésére, ha a gépek száma több, mint kettő, akkor a feladat NP teljes [25].

Ütemezési feladatok jelentős részéről bizonyítható, hogy NP teljesek, azaz más nehéz feladatokkal ekvivalensek. Például az egy gépes növekvő végrehajtási idejű taszkokat tartalmazó súlyozott befejezési idejű ütemezési feladat NP teljességének igazolását találhatjuk Bachman és társainak munkájában [7]. Az NP teljességet az ütemezési feladat egy másik NP teljes (N3P) feladattá transzformálásával igazolták. Nott és Lee egy halmaz lefedési feladattá alakították át az eredeti ütemezési feladatot [62].

Az ütemezési feladatokkal ekvivalens az ún. hátizsák rakodási feladat. A hátizsák rakodási feladatnál adottak az $s_1, s_2, ..., s_m$ súlyok és a súlyokhoz tartozó $v_1, v_2, ..., v_m$ értékek, valamint a hátizsákba rakható megengedett maximális b összsúly. A feladat, hogy találjunk egy olyan $I \subset \{1, 2, ..., m\}$ részhalmazt, melyre a $\sum_{i \in I} s_i \leq b$, ugyanakkor a $\sum_{i \in I} v_i$ a lehető legnagyobb. A hátizsák rakodási feladat NP-teljes.

Láda pakolási feladatoknál (*bin packing*) adottak az $s_1, s_2, ..., s_m$ súlyok, melyek mindegyike $0 \leq s_i \leq 1$ racionális szám, és adott a $k > 0$ egész szám. A feladat annak eldöntése, hogy a tárgyakat bele lehet-e pakolni legfeljebb k számú egységnyi kapacitású ládába. A láda pakolási feladat is NP-teljes. Azar és Regev a klasszikus láda pakolási feladat egy módosítására (*bin stretching*) adtak on-line algoritmust, mely láda pakolási feladatban a súlyok minél egyenletesebb ládákba osztása a cél [4]. Ez a láda pakolási feladat ekvivalens az olyan ütemezési feladatokkal, ahol a terhelés minél jobb szétosztása a cél (*load balancing*).

Általában az ütemezési feladatok felírhatóak egy vegyes egész lineáris/nem lineáris programozási feladatként (*Mixed Integer Linear/Non Linear Programming*, MILP/MINLP). Az ütemezési feladatot MILP/MINLP matematikai programozási modellként felírva majd megoldva, a megoldás megkeresése egy NP nehéz feladat.

2.3. Ütemezési feladatok a szakirodalomban

A szakirodalomban a legtöbb ütemezési feladat a vegyiparból és a műszaki termelő rendszerek kapcsán keletkezik. Az ipari ütemezési feladatokon kívül más területeken is találhatunk ütemezési feladatokat. Amico és Martello bebizonyították, hogy az open shop ütemezési feladat és a műholdon keresztül időosztásos módon kommunikáló földi állomások optimális ütemezésének problémája ekvivalens feladatok [2]. Az SS/TDMA (*Satellite-Switched/Time Division Multiple Access*) feladatban egy műhold segítségével kommunikál több különböző földi állomás. A műhold kapcsolási táblájától függ, hogy mikor melyik két állomás kommunikálhat egymással. A kommunikációhoz szükséges idő a kommunikáció során elküldött információ méretével arányos. Ha adott az egyes állomások kimenő kommunikáció igénye, akkor keressük azt a műhold kapcsolási tábla sorozatot, mellyel a rendszer kommunikációja a legkevesebb idő alatt végbemegy.

Zhang és Bard munkájukban a levélfeldolgozó és szétosztó rendszerek működését vizsgálták [102]. Ezek olyan nagy méretű rendszerek melyek fogadják, rendezik és továbbítják a postai leveleket. A fő probléma a feladat mérete mellett a berendezések és az emberi erőforrások megfelelő összehangolása. A feladat megoldására két módszert is ajánlanak. Az első módszerben relaxálják az ütemezési feladatot egy lineáris programozási modellé (*Linear Programming*, LP), majd az LP modell eredményei alapján építenek fel egy heurisztikus algoritmust. A második megközelítésükben a Benders dekompozícióra alapozva építik fel az algoritmusukat.

Érdekes ütemezési feladatot fogalmaztak meg Arkin és társai egy hivatal működését vizsgálva [3]. A szerzők „perverz” ütemezési feladatnak hívják az ún. lusta bürokraták optimális ütemezési feladatát a bürokraták által használt különleges célfüggvény miatt. A bürokraták célja, hogy minél kevesebb munkát végezzenek el, ne

veszítsék el a munkájukat és mindig legyen valami munkájuk, amire hivatkozva további kellemetlen feladatokat kerülhetnek el. A lusta bürokrata feladat is NP nehéz.

Yu és társai az FMS termelő struktúra ütemezésére a Petri hálókat és a heurisztikus kereső módszereken alapuló mesterséges intelligencia eszközeit használták [99, 100]. A rugalmas termelő rendszer számos berendezést és automatizált munkaanyag továbbító rendszert tartalmaz, mely a munka-anyag megfelelő helyre szállításáért felelős. A rendszer rugalmassága abban rejlik, hogy a termékeket több különböző úton elő lehet állítani. Az ütemező logika határozza meg, hogy melyik terméket milyen berendezés állít elő és milyen időintervallumban. Az ütemezés ábrázolására a Petri hálók új osztályát, a Buffer-hálókat vezették be, mely ábrázolja a feladatosztály speciális tulajdonságait. Az ütemezési architektúra integrálja a Petri hálókat és a mesterséges intelligencia eszközeit. Bevezettek egy új heurisztikát, mely a Petri hálókon alkalmazva drasztikusan csökkenti a keresési teret. Ez a heurisztika az erőforrás elérhetőségi költség mátrixon alapul, mely mátrix pedig a Buffer-háló tulajdonságai alapján építhető fel.

Heilmann munkájában korlátozott erőforrást tartalmazó projekt ütemezési feladatok megoldására adott egy egzakt szétválasztás és korlátozás típusú algoritmust [31]. A projekt olyan ütemezését keresi, mely végrehajtási ideje a lehető legkisebb. Az ütemezéshez a tevékenységek kezdési idejének és végrehajtási módjának meghatározása a feladat. A tevékenységek végrehajtási módtól függően más típusú és mennyiségű erőforrást igényelnek.

Projekt ütemezésére kidolgozott szimulált hűtés és tabu kereső módszereket mutattak be Mika és társai munkájukban [55]. Munkájukban figyelembe veszik a projekt teljesítése közben jelentkező pénzügyi folyamatokat. Az ütemezési feladatok ábrázolására a tevékenység a csomópontban (*Activity on Node*, AoN), vagy a tevékenység az élen (*Activity on Arc*, AoA) típusú gráfokat szokás használni. A publikációban a projektet a tevékenység a csomópontban típusú gráffal ábrázolják a szerzők.

Kondili és társai az STN (*State Task Network*) gráf-reprezentációt vezették be az ütemezési feladatok ábrázolására [41]. Az STN egy páros gráf, mely a műveletek és anyagok kapcsolatát ábrázolja. Az ábrázolás hasonlóságot mutat a Friedler és társai

által korábban publikált P-gráf módszertanra, melyet folytonos műveleteket tartalmazó hálózatszintézis feladatok megoldására vezettek be [20, 21]. Az STN modellben az idő horizont diszkretizálása alapján MILP vagy MINLP matematikai programozási modellt írtak fel, melyet kereskedelmi megoldókkal oldanak meg. Az STN alapú matematikai programozási modelleknél kulcskérdés a diszkretizáció finomsága és módja. Az ekvidisztáns intervallumok alapján felírt MILP modellt „diszkrétnek", a változó hosszú intervallumok alapján felírtakat „folytonos" típusúnak nevezik a szakirodalomban. A folytonos STN alapú MILP modellek az események kezdésének és befejezésének folytonos ábrázolását jelenti, azonban ezekben a folytonos modellekben is csak diszkrét, rögzített számú eseményt kezelnek. A módszer komoly hátránya, hogy a matematikai modell a diszkretizáció finomságától függően vagy feleslegesen sok döntési változót tartalmaz és így a megoldása nehézzé válik, vagy a nem eléggé finom felosztás esetén a döntési változók száma elfogadható, de a modell kizárja az eredeti feladat optimális ütemezésének megtalálását. A matematikai programozási modellekkel a gyártásban jelen levő anyag tárolási korlátozások nehezen, vagy egyáltalán nem kezelhetők. Floudas és Lin összefoglalja, az STN ábrázolást és matematikai programozási modellt használó módszereket, részletesen elemzik a modellekben használt idő ábrázolási módokat [19].

Számos matematikai programozási modell létezik ütemezési feladatok megoldására. A MILP matematikai programozási feladatok [12, 41, 49, 54, 69, 70, 103], vagy MINLP feladatok [58, 81] olyan leszámlálási technikák, melyek elméletileg megadják a modellezett ütemezési feladat optimális megoldását. A gyakorlatban ezeknek modelleknek a megoldása elfogadhatatlan nagy számítási teljesítményt igényel. Lokális keresőkkel, mint például tabu kereső algoritmussal, vagy „szimulált hűtés" (*simulated annealing*) módszerével működő kereséssel kisebb számítási teljesítménnyel megoldhatjuk az ütemezési feladatot, azonban ezen megoldások optimalitása általában nem garantált.

Az STN alapú matematikai modell továbbfejlesztésére sok publikációt találhatunk a szakirodalomban. Nott és Lee egy cukoripari feladat megoldására alkalmazták a módszerüket, és összehasonlították a hagyományos MILP modellek megoldásához szükséges futási időkkel [62]. Arra a következtetésre jutottak, hogy a MILP modellek

[41] megoldásához elfogadhatatlanul nagy memória és processzor idő szükséges már átlagos méretű feladatok megoldásánál is.

Mockus és Reklaitis az STN matematikai modellje alapján egy MINLP modellt írt fel szakaszos feladatok optimális ütemezésére [59]. A nagyméretű matematikai modell megoldására a „Bayes heurisztikát" használták. A heurisztika lényege, hogy egy döntési változó kiválasztásához különböző valószínűségeket rendelve, a kiválasztás lépése a valószínűségek alapján történik a keresési fában.

A szakaszos üzemű berendezéseket időnként tervezett módon, vagy meghibásodás miatt karban kell tartani, illetve javítani kell. Eközben a gyártási folyamatból nyilvánvalóan kiesnek ezek a berendezések. Sanmartí és társai módszert dolgoztak ki az előrelátható karbantartások és a nem várt meghibásodások figyelembevételére [83]. Az STN ábrázolás alapján felírt modellel keresik azt az ütemezést, mely a lehető legrobusztusabb, azaz meghibásodó berendezések kiesésével a gyártási folyamat folytatható.

Nott és Lee szakaszos és folytonos műveleteket is tartalmazó termelési rendszereket vizsgáltak [61]. Amikor a folytonos műveleteket is szakaszos folyamatként ábrázolják, akkor a kapott MILP modell diszkrét változóinak száma jelentősen megnő, a modell bonyolultsága miatt a megoldása igen nehéz. A javasolt módszerben a hagyományos MILP modell használata helyett a modellt hierarchikusan felbontják és kontroll módszerek alkalmazásával hatékonyabban megoldják.

Az STN matematikai programozási modelljének nem egyenlő közű idődiszkretizációra való kiterjesztése található Mockus és Reklaitis munkájában [60]. A megoldandó matematikai programozási modell egy MINLP feladat, ami egyszerűsíthető egy olyan MIBLP (*Mixed Integer Bilinear Programming*) feladattá, mely csak a célfüggvényében nemlineáris. Vizsgálatot végeztek a megoldható feladatok méretére is.

Az STN ábrázolást Pantelides kibővítette és létrehozta az RTN (*Resource Task Network*) gráfot [66]. Az RTN ábrázolás egy olyan STN gráf, melyet kiegészítettek a taszkhoz rendelhető berendezésekkel és erőforrásokkal. Ugyanúgy, mint az STN gráf alapján, az RTN gráf alapján is egy MILP, vagy MINLP matematikai programozási modell írható fel az ütemezési feladat megoldására.

Méndez és társai a szakaszos folyamatok ütemezésére kidolgozott modelleket dolgozták fel és elemezték publikációjukban, elsősorban az STN és RTN ábrázolás alapján létrehozott matematikai modelleket vizsgálták [53]. A különböző gráf-ábrázolási technikák mellett a matematikai programozási modellek a különböző korlátozás típusokban (rögzített/változó batch méret, berendezés váltások, köztes anyag tárolási és továbbítási módok) és célfüggvény típusokban (végrehajtási idő, koraiság, gyártási költség) térnek el egymástól. A megoldandó feladat jellege és a használt matematikai programozási modell együttesen határozza meg a megoldható ütemezési feladat méretét.

Az ellátási láncok menedzsmentje (*Supply Chain Management*, SCM) először az 1990-es évek elején került a figyelem középpontjába. Az ellátási láncnak (*Supply Chain*, SC) az üzleti partnerek hálózatát (beszállítók, gyártók, szétosztók és eladók) nevezzük, amik együtt dolgoznak azon, hogy a nyersanyagokból köztes- és végtermékeket állítsanak elő, majd ezeket eljuttassák a kiskereskedésekbe. A „vegyipari ellátási lánc" a SCM vegyiparra való leszűkítését jelenti. Grossmann és Westerberg a vegyipari SCM-el foglalkoztak munkájukban [27]. Az SCM megpróbálja a gyártást integrálni a beszállítókkal és a vevőkkel oly módon, hogy egy egészként kezeli a teljes rendszert, miközben felügyeli és irányítja a rendszer be– és kimeneteit. Ily módon a termékek megfelelő mennyiségben kerülnek előállításra, és a piaci igényeknek megfelelő módon lesznek szétosztva. Guillén és társai a vegyipari ellátási folyamat tervezés és ütemezésére pénzügyi folyamatokkal integrálva dolgoztak ki egy STN alapú matematikai modellt [28]. Céljuk a vállalati szintű tervezés támogatása. A részfeladatok egymásutáni megoldását összehasonlítva az integrált megoldással igazolták modelljük működését.

Stefanis és társai többcélú ütemezési feladatként kezelik a szakaszos és folytonos műveleteket tartalmazó rendszert, melyben tervezési, ütemezési és környezetszennyezési aspektusokat kezelnek [91]. A folytonos folyamatok környezeti hatásainak jellemzésére az LCA (*Lifecycle Analysis*) módszertant használják. Az algoritmust tejiparból származó ütemezési feladat megoldásával szemléltetik.

Min és Cheng genetikus algoritmust használnak a termelés költségének minimalizálására, miközben teljesíteni kell a termékekhez rendelt határidőket. Szimulált hűtés

és finom heurisztikák segítik, hangolják a genetikus algoritmust, a hatékonyabb megoldás kereséséhez [56].

Majozi és Zhu integrált tervezési és ütemezési feladatok megoldására dolgoztak ki egy matematikai modellt és a modell megoldására egy módszert [49]. A matematikai modell felépítéséhez a fuzzy halmazokat használják a bizonytalan, vagy nem rendelkezésre álló információk ábrázolására. A fuzzy halmazok alapján a matematikai modell egy MILP modellel fogalmazható meg. A MILP modell megoldása az operátorok üzemek közötti optimális elosztását adja.

Dunstall és Wirth összegezték és összehasonlították az ütemezési feladatokra bevezetett szétválasztás és korlátozás típusú algoritmusokat [18]. A több párhuzamos gépet tartalmazó feladatokat vizsgálták, mely feladatok bizonyítottan NP nehezek. A jobokat osztályokba sorolták. Az egy osztályon belüli jobok egymás utána végrehajtásához egy adott gépen nincs beállítási idő (*setup time*). Ha különböző osztályokban szereplő jobokat hajtunk végre egymásután, akkor a két job végrehajtása között meghatározott ideig a gépnek állnia kell. A feladatok nem megszakíthatóak. Munkájukban különböző döntési stratégiákat hasonlítottak össze.

Tang és társai a hibrid flow shop feladat megoldására dolgoztak ki algoritmust [96]. A vizsgált feladatban olyan ütemezést kerestek, melyben a termékek súlyozott gyártási idejének összege minimális. A flow shop feladat megoldására a Lagrange relaxációt használták. Acél gyártásával kapcsolatos ipari feladattal szemléltetik módszerüket.

Azaron és társai egy multi-objektív projekt ütemezési feladatot oldottak meg PERT modellt használva [5]. A rendszer döntési változói a projekt tevékenységekhez rendelhető erőforrások mennyisége. A modellben négy konkurens célfüggvényt használnak.

Szakaszos termelő folyamatokban a berendezéseket köztes tárolóként használva növelhetjük a rendszer termelékenységét, hatékonyságát. Ha és társai bemutattak egy MILP modellt a minimális végrehajtású ütemezés meghatározására figyelembe véve a különböző lehetséges köztes anyag tárolási politikákat [30]. Munkájukban az NIS, FIS, UIS, ZW tárolási politikák ismertették.

Sarker és Yu szakaszos üzemű flow shop feladatokhoz az optimális batch méretét keresi a minimális költségű ütemezéshez [87]. A költségfüggvény három komponenst

tartalmaz, a köztes anyagok és a termékek tárolási költségeit és a berendezések működési, konfigurációs költségeit. Két heurisztikus algoritmus segítségével adja meg a költséget és a hozzá tartozó ütemezést.

Sok esetben szükséges (pl. egy új megrendelés teljesítésének határidejének kialakításakor), hogy gyorsan megkapjuk, vagy megbecsüljük egy taszk-halmaznak a várható végrehajtási idejét anélkül, hogy meghatároznánk a hozzá tartozó pontos ütemezést. Raaymakers és Fransoo statisztikai elemzéssel és regressziós analízissel becslő eljárást dolgozott ki a várható végrehajtási idő gyors meghatározására [73].

Raaymakers és Hoogeveen a végrehajtási idő becslésére szimulált hűtés módszerét használja [72]. A vizsgált várakozás mentes job shop ütemezési feladat NP nehéz, így nehéz hatékony és optimalitást garantáló algoritmust találni megoldásukra. A szimulált hűtés lokális keresésen alapuló hatékony optimalizációs módszer, mely véletlenszerű szomszédsági keresés elvén működve valamilyen valószínűséggel fogad el új megoldásokat. Ha az algoritmus egy olyan lokális optimumot talál, mely nem globális optimuma a feladatnak, akkor az algoritmus megpróbál kiszabadulni a megtalált lokális optimum környezetéből, a globális optimum megtalálásának a reményében.

Szakaszos és félszakaszos üzemek gyakran állandó mennyiségű termékeket állítanak elő minden vizsgált időintervallumban. Általában érdemes egy olyan periodikusan ismétlődő ütemezést meghatározni, melyet az időintervallumokban egymás után végrehajtunk. A periodikusan ismétlődő ütemezés mellett a periódus optimális hosszának a meghatározása is feladat. Periodikus ütemezésnél a cél egy időperiódus optimális ütemezésének a meghatározása, általában a „beindító" és „leállító" periódust figyelmen kívül hagyva. Schilling és Pantelides szakaszos periodikus ütemezési feladatok megoldására dolgoztak ki egy algoritmust [88, 89]. A feladatosztály ábrázolására az RTN gráfot használják. A feladatosztály megoldására a folytonos időábrázolást használó MINLP modellt írták fel. A szerzők egy speciális szétválasztás és korlátozás elvű algoritmussal oldjak meg a MINLP modellt.

Reaktív ütemezési feladatok esetén (*reactive scheduling*) a folyamatban levő, végrehajtás alatt álló ütemezés folytatása valamilyen körülmény megváltozása miatt akadályba ütközik. Például ha egy beütemezett rendelés megszűnik, vagy egy új rövid határidős munka jelentkezik, illetve ha például egy berendezés tönkremegy, vagy egy

munkás megbetegszik. Ilyen esetekben a reaktív ütemezésnek a feladata, hogy a futó ütemezést oly módon megváltoztassa, hogy befejezhető legyen és az új igények is ki legyenek elégítve [93].

Ütemezési feladatok nagy részében feltételezik, hogy a taszkok végrehajtási idejei determinisztikusak. Egy valódi vegyipari folyamatban azonban ezek az idők nagyrészt bizonytalanok. Honkomp és társai a bizonytalan végrehajtási idővel működő berendezések ütemezésére írtak fel STN alapú matematikai programozási modellt [36]. A modellt egyenlő és változó közű idő diszkretizálás esetén is megoldották. A feladatnak a robusztus megoldását keresték, mely ütemezés a végrehajtási idők változásaira a legkevésbé érzékeny. A sztochasztikus jellegű szakaszos folyamatok modellezését és optimalizálását dolgozták ki munkájukban [37]. Az optimalizáló algoritmus és az ütemező szimulátor összekapcsolásával vizsgálták a sztochasztikus folyamatok viselkedését.

Honkomp és társai összegyűjtötték az ütemezési feladatok során jelentkező fontos gyakorlati (ipari) és elméleti (akadémiai) szempontokat [35]. A feladat definíciója, a feladat mérete, az ütemezendő berendezések típusai, a termékek és köztes anyagok tárolásának módjai, az ütemezéshez kapcsolódó egyéb tevékenységek, a termelő folyamat során használt gyártási technológia és az operátorok rugalmassága határozza meg a feladat megoldásának menetét.

Puigjaner és Espuna az egész termelési folyamat modellezésére és kezelésére adtak integrált megoldást [71]. Munkájukban ábrázolják és részletesen leírják a termelő folyamatokat. Támogatást adtak a tervezési és ütemezik kérdésekhez kapcsolódó döntésekhez. Ellenőrzik és irányítják a rendszer termelési folyamatelemeit.

Bank és Werner a különböző időpontokban jelentkező feladatok ütemezésére használtak heurisztikán és lokális keresésen alapuló algoritmust [8]. A feladatban a jobokat közös határidőre kell végrehajtani a berendezéseken. Az algoritmus a feladatokat megpróbálja úgy ütemezni, hogy a határidőtől való eltérés súlyozott összege minimális legyen.

Henning és Cerdá a matematikai programozási modellek helyett a diszkrét eseményű rendszerek és a mesterséges intelligencia területét és elsősorban a tudásbázist használja szakaszos folyamatok ütemezésére [32]. Munkájukban definiálják szakaszos

folyamatok ütemezése során jelentkező fogalmakat (termék, rendelés, kampány, batch, működés, taszk, berendezés, erőforrás, ütemezés).

Méndez és Cerdá egy speciális gyártási folyamat matematikai modelljét adták meg, mely a feladat optimális ütemezés szolgáltatja [51]. A gyártási folyamat két fázisból áll: gyártó fázis több párhuzamos berendezéssel, majd a köztes anyag tárolás tartályokban. A termékek gyártására és tárolására használható berendezések rögzítettek, továbbá a berendezések topológiai elrendezése is korlátozza az ütemezést. A feladat megoldására a szerzők egy folytonos idejű MILP modellt használnak kiegészítve a berendezések taszk sorrend függő váltási idővel és a termékek különböző szállítási határidővel.

Subramanian és társai kutatási fejlesztési projekt irányítására egy sztochasztikus optimalizálási modellt vezettek be [92]. A cikkben Sim-Opt architektúrát alkalmazva szimulációs és optimalizálási lépéseket hajt végre a feladaton. A szimulációt diszkrét eseményű rendszer segítségével valósítják meg, az optimalizáláshoz matematikai programozási modellt írnak fel és oldanak meg.

Általában a job shop ütemezési feladatokban feltételezzük, hogy minden mennyiség, így taszkok végrehajtási idői is, rögzített, determinisztikus mennyiségek. Ez a feltételezés akkor tekinthető jónak, ha a vizsgált folyamat teljesen automatizált. Ha a folyamatban szerepelnek emberi beavatkozások is, akkor az ütemezési feladat sztochasztikus modellekkel pontosabban kezelhető. Ghrayeb munkájában fuzzy job shop ütemezési feladatot publikált [23]. A modell egy többcélú optimalizálási feladat, melyben az ütemezés végrehajtási idejének szórás értéke és az ütemezés végrehajtási ideje szerepel. A bizonytalan taszk végrehajtási időket fuzzy logikával kezeli. A modellt genetikus algoritmust alkalmazva oldotta meg a szerző.

Chan és Swarnkar rugalmas termelő rendszerek ütemezésére a hangya algoritmust alkalmazták [13]. Az algoritmus működése a hangyák viselkedését követi. A mesterséges hangyakolónia-rendszerekben (*Ant Colony Systems*, ACS) a hangyák azon képességét használják ki, hogy a lehetséges útvonalak közül rátalálnak a legrövidebbre útra, miközben majdnem teljesen vakok. A hangyák „látását” egy anyagnak, a feromonnak köszönhetik. A hangyák változó mennyiségű feromont hagynak útvonalukon. A hangyák valamilyen valószínűség alapján követik a többi hangya feromon jeleit. A

biológiai rendszer ezen érdekes tulajdonságait átültetve lokális optimumot adó kedvező tulajdonságú algoritmust kapunk.

Jayaraman és társai szakaszos folyamatok ütemezését és az ütemezés során a köztes anyag tárolásának lehetséges eseteit elemzik. A hangya algoritmust alkalmazták batch folyamatok hatékony megoldására [39]. Rajendran és Ziegler flow shop ütemezési feladatok megoldására mutattak be hangya algoritmust publikációjukban [77].

Sanmartí és társai szakaszos sztochasztikus folyamat ütemezésére dolgoztak ki egy módszert [85]. A módszerben a bizonytalanság a berendezések működési idejében rejlik. Olyan „robusztus" ütemezéseket keresnek, melyek bizonytalan környezetben is megfelelően végrehajtható. A [82] munkában a bizonytalanság a berendezések meghibásodásából adódik. On-line adatbázisok segítségével megelőző karbantartások ütemezésével csökkentik a berendezés meghibásodások miatti rendszer leállásokat.

Nagy mennyiségű termék határidőre való gyártásánál fontos lehet a határidők minél pontosabb betartása a tárolási költségek csökkentése miatt. Ohta és Nakatani [63] heurisztikus módszereket vezetett be a tárolási költségeket figyelembe vevő ütemezés meghatározására. Az ütemezési feladatot diszjunktív gráffal ábrázolta, mely gráfban a leghosszabb út kereső algoritmus segítségével határozza meg a taszkok határidőhöz képest való késését vagy sietését.

Az értekezés további fejezeteiben szakaszos ütemezési feladatok optimális megoldására mutatok be módszereket. Az ütemezési feladatokban az NIS tárolási stratégiát feltételezem. A bemutatott algoritmusok kihasználják a gráf ábrázolás segítségével a feladatok kombinatorikus tulajdonságait. A feladatok optimális megoldását szétválasztás és korlátozás elvén működő algoritmusokkal határozom meg.

3. fejezet

S-gráf módszertan bemutatása

Az *S-gráf módszertan* (*S-graph framework*) [84, 86] szakaszos folyamatok optimális ütemezésének meghatározására bevezetett gráf ábrázolási mód és hatékony algoritmus. A módszertant az általános ütemezési feladatok megoldására hozták létre, azonban a szerzők lehetőséget biztosítottak speciális ütemezési feladatok S-gráf módszertannal történő megoldására, az alap módszertan feladatfüggő gyorsítási lehetőségeire.

Az értekezés következő fejezeteiben egy-egy speciális ütemezési feladatosztályt mutatok be és oldok meg. Mindegyik feladatosztály megoldásához az S-gráf módszertant használom, vagy ebből a módszertanból kiindulva új algoritmusokat hozok létre szakaszos ütemezési feladatokhoz kapcsolódó más feladatosztályok megoldására. Ezért ebben a fejezetben az eredeti S-gráf módszertan bemutatása és működésének, hatékonyságának az illusztrálása a célom.

A szakirodalomban sokfajta ütemezési feladattal találkozunk, melyek jellegükben, bonyolultságukban, a megoldásukra kidolgozott módszerekben jelentősen különböznek egymástól. Az S-gráf módszertanban az ütemezési feladatoknak azt az osztályát tekintjük, melyben a cél egy olyan optimális ütemezés megtalálása, mely képes a lehető legrövidebb idő alatt a kívánt mennyiségű termékek előállítására a rendelkezésre álló szakaszos üzemű *berendezések* felhasználásával. A kívánt termékmennyiség előállításának teljes idejét végrehajtási időnek nevezzük. Minden termék taszkok rögzített sorrendű hálózatával állítható elő. A gyakorlatban egy taszk általában több alternatív berendezéssel végrehajtható (nem feltétlenül ugyanannyi idő alatt). Az

ütemezési algoritmus feladata ezen berendezések közül minden taszkhoz egy megfelelőt hozzárendelni és a berendezésekhez rendelt taszkok végrehajtási sorrendjét, azaz a berendezések ütemezését a meghatározni.

Egy ütemezési feladat megoldását, azaz a berendezésekhez tartozó taszkok ütemezését egyértelműen megadhatjuk a berendezésekhez rendelt taszkokon megadott rendezés formájában. Ha a feladat megoldásában egy berendezéshez az i, j és k jelű taszkokat rendeltük hozzá, és a berendezés a felsorolás sorrendjében hajtja végre ezeket a taszkokat, akkor ezt a taszk végrehajtási sorrendet nevezzük a *berendezés ütemezésének*. Az ütemezési algoritmus célja a berendezések olyan ütemezésének a meghatározása, mely a feladat optimális ütemezését adja.

3.1. Szakaszos ütemezési feladat megadása

Többcélú (*multipurpose*) szakaszos jellegű ütemezési feladatok megadhatóak a termékekhez tartozó receptekkel, a receptekben szereplő taszkokhoz hozzárendelhető berendezések halmazaival és az előállítandó termékek mennyiségével. A recept egy olyan dokumentum, mely minimálisan tartalmazza az adott termék gyártásához szükséges adatokat. Az ISA SP88 szabvány négy szintjét definiálja a recepteknek. Ezek a szintek a következők:

- Az általános recept (*general recipe*) a gyártásban szereplő nyersanyagokat, termékeket és az anyagokhoz tartozó mennyiségeket tartalmazza. Az általános recept azonban nem tartalmazza a hely specifikus recept információkat.

- A hely recept (*site recipe*) az általános recept hely specifikus információkkal kibővített módozata.

- A mester recept (*master recipe*) már a berendezés függő információkat is tartalmazza. Így például tartalmazza a berendezések működési idejét. Továbbá tartalmazza a rendelkezésre álló nyersanyagok mennyiségét és a termékek előállításának a folyamatát.

- A kontroll recept (*control recipe*) a mester receptből származó recept, mely további információkat tartalmaz a berendezések működtetéséről.

A négy recepttípus közül a mester receptet használhatjuk szakaszos folyamatok leírásához. Továbbiakban ebben a dolgozatban ezt a recept típust használom ütemezési feladatok megadására és receptként hivatkozok rá.

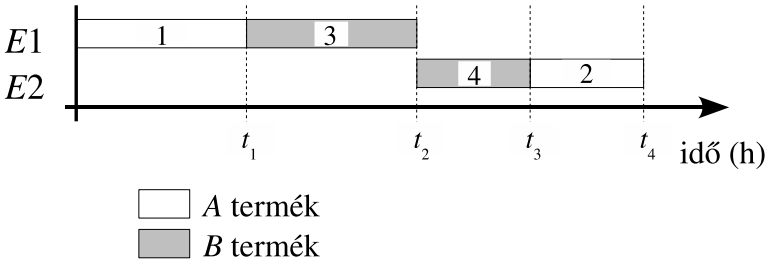
Egyszerű receptnek nevezzük az olyan gyártási folyamatot, melyben az adott termékhez tartozó taszkokat szekvenciálisan hajtjuk végre, azaz nincsen a receptben olyan taszk, mely kimeneteit több taszk használja fel, és nincs olyan se, melynek több bemenete különböző taszkoktól származik. Összetett a recept, ha van benne olyan taszk, melynek több közvetlen megelőző taszkja van, vagy mely után több taszkot is végre lehet hajtani a recept alapján. Szemléletesen, egyszerű recept nem tartalmazhat elágazást vagy csomópontot, összetett recept tartalmazhat.

A gyártási folyamatban megkövetelt tárolási stratégiának hatása van az ütemezési feladat megoldására és a feladat bonyolultságára. A tárolási stratégia befolyásolja az ütemezés megvalósíthatóságát. Ebben a részben az UIS és az NIS stratégiának az ütemezés megvalósíthatóságra való hatását mutatom be.

Egy ütemezést megvalósíthatónak nevezzük, ha az ütemezés alapján a gyártási folyamat végrehajtható. A megvalósítható ütemezés (*feasible schedule*) gondoskodik a köztes anyagok tárolási stratégiának megfelelő kezeléséről, biztosítja, hogy a berendezések egy időben csak egy taszkot hajtanak végre, minden taszkot végrehajt egy megfelelő berendezés, továbbá az ütemezés figyelembe veszi a berendezések új taszk elkezdéséhez szükséges váltási idejét és a taszkok között fennálló precedenciákat.

Már az ütemezési stratégiák elnevezései is sugallják, hogy az NIS stratégia esetén nehezebb megvalósítható ütemezést találni, hiszen az NIS stratégia alkalmazása esetén kényszerként jelen van a köztes anyagok tárolásának biztosítása a berendezések segítségével. Egy példán keresztül szemléltetem, hogy van olyan ütemezés, mely megvalósítható UIS stratégia esetén, ugyanakkor nem megvalósítható az NIS stratégia esetén.

Az ütemezések leírására megfelelő grafikus eszköz a *Gantt-diagramm*. A diagrammon a függőleges tengely mentén ábrázoljuk az ütemezett berendezéseket, a vízszintes

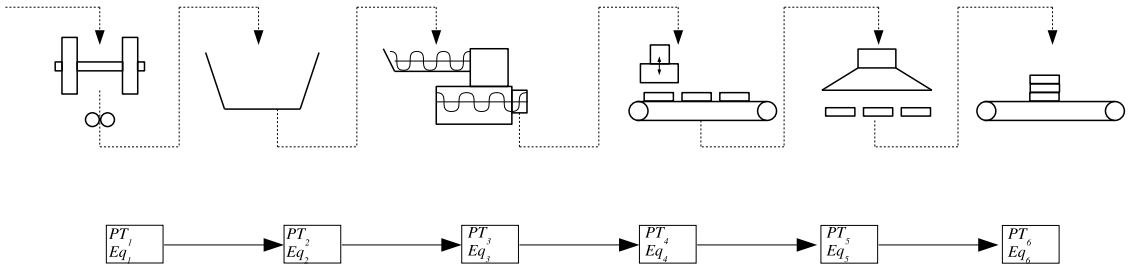


3.1. ábra. Ütemezés ábrázolása Gantt-diagrammal.

tengely pedig az idő tengely. Ha egy berendezés két időpont között végrehajt egy hozzárendelt taszkot, akkor ezt a diagrammon a berendezés sorában szereplő időpontok közé rajzolt téglalappal jelöljük a diagrammon. A Gantt-diagrammokon az ugyanahhoz a termékhez tartozó taszkokat szokás a téglalapok színezésével, kitöltésével jelölni.

A 3.1 ábra Gantt-diagrammja az $E1$ és $E2$ berendezés egy lehetséges ütemezését mutatja. Az A termék előállításához az 1, majd a 2-es taszkot kell végrehajtani, a B termék előállításához először a 3-as, majd a 4-es taszkot kell végrehajtani. A Gantt-diagrammon látható ütemezés alapján az $E1$ berendezés az először az 1-es, majd a 3-as taszkot, az $E2$ berendezés pedig először a 4-es, majd a 2-es taszkot végzi.

A 3.1 ábra ütemezése, olyan ütemezés, ami NIS stratégia esetén nem megvalósítható, UIS esetén azonban megvalósítható. Az ütemezés alapján az $E1$ berendezés először végrehajtja az 1-es taszkot. Ha végzett az 1-es taszkkal, akkor az NIS stratégia esetén a köztes anyagot áttölti a recept alapján a következő taszkhoz ütemezett berendezésbe, az $E2$ berendezésbe. Ezután $E1$ berendezés végrehajtja a következő hozzárendelt taszkot, a 3-as taszkot. NIS stratégia esetén nem lehet folytatni a Gantt-diagramm ütemezése alapján a gyártási folyamatot, hiszen az $E2$ berendezés a benne tárolt köztes anyag miatt nem tudja elkezdeni a hozzá ütemezett első taszkot. Ugyanez az ütemezés UIS esetén megvalósítható, mert ennél a stratégiánál nem a berendezések tárolják köztes anyagokat, így az ütemezéshez tartozó gyártási folyamat végrehajtható, megvalósítható.



3.2. ábra. Téglagyártás receptje.

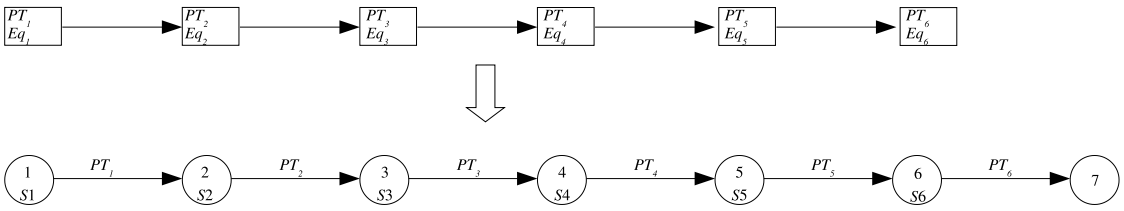
3.2. Szakaszos folyamatok ábrázolása S-gráffal

Az S-gráf ábrázolás szakaszos, többcélú ütemezési feladatok leírására alkalmas. Megfelelően ábrázolja a feladatok és ütemezések strukturális tulajdonságait. Az S-gráf ábrázolás az NIS stratégiához készült, azonban bármelyik előzőleg bemutatott tárolási stratégiához is használható.

A terméket előállító recept hagyományosan ábrázolható egy olyan irányított gráffal, amelyben a csúcsok ábrázolják a termelési folyamat taszkjait és a taszkok között levő élek pedig a taszkok sorrendjét (taszkok közötti függőségeket) jelentik. A taszkokhoz rendelhető berendezések halmaza és a berendezések működési idői a recept megfelelő csúcsaiban adottak.

Példaként vegyük a téglagyártás egyszerűsített folyamatát. A téglagyártás receptje alapján a következő műveleteket hajtják végre: nyersanyag aprítása, a nyersanyag vízzel keverése, vákuum csigaprés használata, az anyagszalag méretre vágása, rakat képzés és kemencében a rakatok kisütése, a tégla csomagolása. A 3.2 ábrán látható a tégla gyártásának egyszerűsített folyamata és a folyamatot leíró irányított gráf. A gráfban a PT_i jelöli a csúcshoz tartozó taszk működési idejét, az Eq_i $(i = 1, 2, ..., 6)$ pedig a taszkot végrehajtható berendezések halmazát. Ebben a példában feltételezzük, hogy a taszk működési ideje nem függ a taszkot végrehajtó berendezéstől.

A recept hagyományos gráfleírásában a csúcsok jelölik a recept taszkjait, az élek mutatják a taszkok sorrendjét. A csúcsok tartalmazzák a taszkok működési idejét és a taszkhoz felhasználható berendezések halmazát. Alakítsuk át ezt a gráfot egy olyan

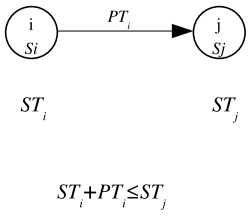


3.3. ábra. Hagyományos gráf átalakítása kombinatorikus algoritmusok számára.

irányított gráffá, mely tartalmazza ugyanezeket a csúcsokat és éleket, továbbá rendeljünk egy-egy új csúcsot a gyártási folyamat során létrejövő termékekhez. Ezekbe a termékeket jelölő csúcsokba mutassanak élek a termékeket gyártó taszkokhoz tartozó gráf csúcsokból. Nevezzük a taszkokhoz tartozó csúcsokat *taszk-csúcsnak* (*task node*), a termékekhez tartozó csúcsokat pedig *termék-csúcsoknak* (*product node*). A gráfban a taszk-csúcsokból kimenő élek értéke a csúcshoz hozzá tartozó taszk működési idejével egyezik meg. Ha a taszk működési ideje a hozzá rendelt berendezés függvényében változhat, akkor az él értéke a legkisebb működési idővel egyezik meg. Végezetül rendeljünk a csúcsokhoz egy egyértelmű azonosítót (pozitív egész szám), illetve jelöljük a taszk-csúcsokhoz rendelhető berendezések halmazát a megfelelő taszk-csúcsokban ($Si = Eq_i$, $i = 1, 2, ..., 6$). A 3.3 ábra mutatja a téglagyártás hagyományos és átalakított gráfját.

Az átalakított gráfban az élek a taszk sorrend jelölésen kívül a gráfban szereplő taszkok lehetséges kezdési ideit is meghatározzák. Az él értéke alsó korlátot jelent a hozzá kapcsolódó két taszk kezdési idejének különbségére, ha az él két taszk-csúcs között található. Ha egy taszk és egy termék-csúcs között található, akkor az él értéke alsó korlát a termék elkészülésének és a taszk kezdési idejének a különbségére. Jelölje ST_i az i, ST_j a j csúcs kezdési idejét. A 3.4 ábrán található gráfrészletre $ST_i + PT_i \leq ST_j$ egyenlőtlenség teljesül a taszkok kezdési idejére.

Az élek jelentése alapján az S-gráfok nem tartalmazhatnak irányított kört. Ha az S-gráf irányított kört tartalmaz, akkor a körben szereplő taszkokra nem adható meg egy egyértelmű rendezés. Ha csak pozitív éleket tartalmazó gráf tartalmazza az irányított kört, akkor egy ellentmondásos egyenlőtlenségrendszert kapunk a körben



3.4. ábra. Élek jelentése az S-gráfban.

szereplő taszkok kezdési ideire. Ha az irányított körben szereplő élek összege nulla, akkor az élek alapján felírt egyenlőtlenség rendszer ugyan nem ellentmondásos (az egyenlőség teljesül), azonban az NIS gyártási folyamat nem hajtható végre, mert a gyártási folyamat köztes anyagainak tárolása nem biztosított. Az ilyen jellegű irányított körök matematikai programozási módszerekkel nehezen kezelhetőek, az S-gráfon végrehajtott körkereséssel azonban könnyen kiszűrhető.

3.3. S-gráf matematikai leírása [86]

Egy irányított G gráfot megadhatjuk az (N, A) párral, ahol az N halmaz a csúcsok véges halmaza, az $A \subseteq N \times N$ halmaz pedig az élek halmaza. Az S-gráf olyan irányított gráf, melynek két típusú éle van, az A_1 és az A_2 halmaz beli élek. Az $A_1 \subseteq N \times N$ halmaz a gráf úgynevezett recept-éleit (*recipe-arc*), az $A_2 \subseteq N \times N$ halmaz a gráf ütemezési-éleit (*schedule-arc*) tartalmazza, továbbá teljesül, hogy $A_1 \cap A_2 = \emptyset$. Bármely $(i, j) \in A_1 \cup A_2$ élhez tartozik egy nemnegatív $c(i, j)$ érték, az él súlya. Tehát a G S-gráf megadható egy (N, A_1, A_2) hármassal (továbbiakban röviden $G(N, A_1, A_2)$).

3.3.1. Recept-gráf

A recept megadja a feladat minden előállítandó termékéhez a gyártandó mennyiséget, a végrehajtandó taszkok sorrendjét, a taszkok között lévő anyagáramokat, és a taszkokhoz hozzárendelhető lehetséges berendezéseket. A recept-gráfot (*recipe-graph*)

a receptből származtatjuk oly módon, hogy létrehozzuk a taszkok kapcsolódását leíró taszk-hálózatot, majd a taszk-hálózatot kiegészítjük a taszkokat végrehajtható berendezések halmazaival.

A termékek gyártását leíró recept alapján a taszk-hálózat felépítéséhez a következő lépéseket hajtjuk végre:

1. Rendeljünk egy-egy gráf csúcsot minden recept beli taszkhoz, illetve egy-egy termék-csúcsot minden termékhez.

2. Mutasson egy él (recept-él) minden taszk-csúcsból a recept alapján utána következő taszk-csúcsba, illetve a terméket gyártó taszkhoz tartozó csúcsból a megfelelő termék-csúcsba.

3. A recept-él súlya legyen egyenlő a kezdő csúcsához tartozó taszk legkisebb működésű idejével.

4. Ha egy termékből nagyobb mennyiségre van szükség, mint amennyi a recept alapján egyszerre elő lehet állítani, akkor a termék előállításában részt vevő taszk- és termék-csúcsokat többszörözzük meg addig, míg a kívánt termékmennyiséget elő nem állítjuk.

Az ilyen módon létrehozott gráfot taszk-hálózatnak nevezzük. Jelölje N_t a taszk-csúcsok, N_p pedig a termék-csúcsok halmazát ($N_t \cap N_p = \emptyset$).

Vannak olyan gyártási folyamatok, melyeket leíró receptek recirkulációt tartalmaznak. Ha egy ilyen recirkulációt tartalmazó gyártási folyamatnak származtatjuk a recept-gráfját, akkor a recirkuláció látszólag egy irányított kört eredményez a taszk-hálózatban. Egy konkrét gyártási folyamatban megvizsgálva a recirkulációban részt vevő taszkok közötti anyagáramokat észrevehetjük, hogy a recirkulációt okozó, visszafelé irányuló anyagáramok nem az aktuális batchhez tartozó termék legyártásához szükségesek, hanem egy későbbi batchből keletkező termékhez. Ezért a receptből származó recirkuláció kezelhető oly módon, hogy a recirkulációt a termékhez tartozó későbbi batch megfelelő taszkjához, mint betáplálás, jelenítjük meg.

A taszk-hálózatból hozzuk létre a recept-gráfot. A $G(N, A_1, A_2)$ recept-gráf olyan S-gráf, mely körmentes és nem tartalmaz ütemezési-éleket, azaz $A_2 = \emptyset$. Legyen E

halmaz a folyamatban szereplő berendezések halmaza. Jelölje N_i az $i \in E$ berendezéssel végrehajtható taszkokhoz tartozó gráf csúcsok halmazát ($N_i \subseteq N_t$). Ahhoz, hogy minden taszkot végre lehessen hajtani feltételezhetjük, hogy $N_t = \bigcup_{i \in E} N_i$, azaz minden N_t-beli taszkhoz találhatunk legalább egy berendezést, mellyel végre lehet hajtani.

Ha a receptben vannak olyan taszkok, melyek bemenő anyagáramai rögzített sorrendben, időzítéssel kell a taszkot végrehajtó berendezésbe jutniuk, akkor a recept-gráf az úgynevezett „betáplálási-sorrend gráffal" kiegészítve biztosítja a bemenő anyagáramok közötti sorrendiséget. A „betáplálási-sorrend gráfról" további részleteket a [33, 86] munkákban találhatunk.

3.3.2. Ütemezési-gráf

Az ütemezési-gráf (*schedule-graph*) olyan S-gráf, mely a feladat megoldását, azaz az ütemezését írja le egyértelműen. A $G'(N, A_1, A_2)$ S-gráfot a $G(N, A_1, \emptyset)$ recept-gráfból származtatott ütemezési-gráfnak nevezzük, ha minden csúcs „ütemezve van". Az ütemezési-gráf olyan ütemezést ír le, mely az NIS stratégia esetén biztosítja a termelés végrehajtását.

Jelölje M_i azoknak a G' ütemezés-gráfbeli csúcsoknak a halmazát, melyhez tartozó taszkokat az $i \in E$ berendezéssel hajtjuk végre. Feltételezzük, hogy a megoldásban olyan ütemezéseket keresünk, melyben minden taszkot pontosan egy berendezés hajt végre, azaz teljesül, hogy $M_i \cap M_j = \emptyset$, ha $i \neq j$ és $i, j \in E$. Ekkor az M_i halmazok ($i \in E$) az N_t halmaznak egy particionálását adják.

Komponens-gráfoknak nevezzük a G' ütemezési-gráf azon részgráfjait, melyek az egyes berendezések ütemezéseit szemléltetik. Az $i \in E$ berendezés G' ütemezéshez tartozó komponens-gráfját jelöljük $G'_i(N'_i, A_{1i}, A_{2i})$-vel, ahol

- N'_i halmaz tartalmazza G'-ből az összes M_i-beli csomópontot és az összes olyan csomópontot, amelybe M_i-ből induló recept-él vezet, azaz
  $N'_i = M_i \cup \{k : k \in N$ és $\exists j \in M_i$, hogy $(j, k) \in A_1\}$.

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

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

- A_{2i} halmaz tartalmazza az összes G'-beli ütemezési-élet, amely az A_{1i} halmaz valamely élének végpontjából az M_i halmaz valamely elemébe mutat, azaz $A_{2i} = \{(j,k) : (j,k) \in A_2, k \in M_i \text{ és } \exists l \in M_i, \text{hogy } (l,j) \in A_{1i}\}$.

A komponens-gráfra teljesül, hogy $G'_i(N'_i, A_{1i}, A_{2i}) \subseteq G'(N, A_1, A_2)$.

A komponens-gráfokban megadott A_{2i} halmazok ütemezési-élei biztosítják ($i \in E$), hogy a köztes anyagok az NIS stratégiának megfelelően a gyártás során folyamatosan valamelyik berendezésben tárolva lesznek. Az ütemezési-él ily módon való behúzásával a berendezés akkor kezdheti el a hozzá ütemezett következő taszk végrehajtását, ha a berendezésben tárolt köztes anyag áttöltésre került a recept alapján következő taszkot végrehajtó berendezésbe. Az A. függelék egy példán keresztül szemlélteti egy ütemezési-gráfhoz tartozó komponens gráfokat.

A [33, 86] munkákban definiálták azokat a tulajdonságokat, melyeket teljesítenie kell egy S-gráfnak ahhoz, hogy a feladatnak egy megvalósítható ütemezését írja le. Ezeket a tulajdonságokat a szerzők négy feltétel formájában fogalmazták meg. Formálisan a következő tulajdonságokat kell teljesítenie a $G(N, A_1, \emptyset)$ recept-gráfból származtatott $G'(N, A_1, A_2)$ ütemezési-gráfnak ahhoz, hogy a feladat egy ütemezését írja le.

(SG1) A G' S-gráf nem tartalmaz irányított kört.

(SG2) A G'_i komponens gráf által definiált 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 ütemezési-él indul.

(SG4) A G' ütemezési-gráf megegyezik komponens gráfjainak uniójával, azaz $G' = \bigcup_{i=1}^{n} G'_i$.

Az S-gráf éleihez rendelt egyenlőtlenségi feltételek a kezdési időkre megkövetelik, hogy az (SG1) feltételnek teljesülnie kell bármely megvalósítható ütemezésre NIS

stratégia esetén. Az (SG2) feltétel az ütemezés teljességét biztosítja, azaz bármely berendezéshez rendelt végrehajtandó taszkok sorrendje egyértelmű. Az (SG3) és (SG4) feltételek nem szükségesek a megvalósíthatósághoz, de belátható, hogy az optimális megoldás mindig megtalálható az általuk leszűkített keresési térben. Ezek a feltételek a nem szükséges és a redundáns ütemezési-élek kiszűréséhez kellenek.

3.4. S-gráf alapalgoritmus ütemezési feladatok megoldására

Az S-gráf alapalgoritmus (a továbbiakban EQ-SG algoritmus) általánosan alkalmazható bármely olyan ütemezési feladatra, melynél a cél a receptből az optimális (minimális végrehajtási idejű) ütemezéshez tartozó ütemezési-gráf meghatározása. Az algoritmus a szétválasztás és korlátozás elvén működik. Ha a feladatnak van megoldása, akkor az algoritmus garantálja a feladat egy optimális megoldásának megtalálását.

Az algoritmus publikálásánál a könnyebb szemléltethetőség céljából a szerzők feltételezték, hogy a recept-gráfban minden taszk pontosan egy darab berendezéssel hajtható végre [86]. Értekezésemben az általános ütemezési feladat megoldására alkalmas algoritmust ismertetem, mely feladatban alternatív berendezésekből kell kiválasztani a taszkokat végrehajtó berendezést és megadni a berendezések optimális ütemezését, azaz a recept-gráfban $|Si| \geq 1$ bármely $i \in N$-re. Az EQ-SG algoritmust a 4. fejezetben bemutatásra kerülő TA-SG algoritmus alapján újra formalizáltam.

3.4.1. Részfeladat ábrázolása és adatstruktúrák inicializálása

A szétválasztás és korlátozás elvén működő algoritmusoknál minden részfeladatban tárolni kell a részfeladat generálásához vezető döntések eredményeit. A döntések ábrázolhatóak egy S-gráfban, hiszen az S-gráf ütemezési-éleit elemezve meghatározhatjuk a berendezések aktuális ütemezéseit, az ütemezésre váró taszkok halmazát, vagy a leghosszabb út kereső algoritmussal az ütemezés végrehajtási idejét. A hatékonyság érdekében ezeket az információkat érdemes a részfeladatokban az S-gráf mellett párhuzamosan tárolni.

A PP részfeladatot a $(G(N, A_1, A_2), bound, last\_node, SOUN, NS)$ ötös definiálja. A részfeladathoz tartozó ütemezést a részfeladat $G(N, A_1, A_2)$ S-gráfja írja le. A $G(N, A_1, A_2)$ S-gráf leghosszabb útjának értéke a *bound*-al egyezik meg. A *last_node* halmaz a berendezések utolsónak ütemezett taszkjait tartalmazza. Ha az $(e, j) \in last\_node$, akkor az $e \in E$ berendezés utoljára a j taszkot hajtja végre. A $SOUN$ halmaz a még nem ütemezett taszkokat tartalmazza. Az NS halmaz a berendezésekhez hozzárendelhető taszkok halmazait tartalmazza. Ha $(e, H) \in NS$, akkor az $e \in E$ berendezéssel a $H \subset N$ halmaz taszkjait lehet végrehajtani. Természetesen minden részfeladatra a $SOUN = \bigcup_{e \in E} NS_e$ egyenlőség teljesül, ezért a $SOUN$ halmaz felesleges, csak a könnyebb implementáció miatt szükséges.

Az EQ-SG algoritmus az *EQ-SG* eljárással indul (lásd 3.5. ábra). Az EQ-SG algoritmus bemenete az ütemezési feladat $G(N, A_1, \emptyset)$ recept-gráfja. A recept-gráfból meghatározhatóak az N_e halmazok ($e \in E$). Az N_e halmaz az e berendezéssel végrehajtható taszkokat tartalmazza. A SET halmaz tartalmazza az EQ-SG algoritmus nyitott részfeladatait, mely halmaz az algoritmus indulásakor üres halmaz. A *current_best* változó tartalmazza a legjobb megoldás értékét, a *solution* változó pedig a hozzá tartozó ütemezést. A *current_best* változó értéke végtelen, amíg az algoritmus nem talált megvalósítható ütemezést a feladathoz.

Az *EQ-SG* eljárás létrehozza a keresési fa gyökér csúcsához tartozó PP részfeladatot. A részfeladat S-gráfja a recept-gráf. A last_node(PP) halmaza üres halmaz, mert még egyetlen taszk sincs ütemezve. A SOUN(PP) halmaz tartalmazza az összes olyan taszkot, melyet valamely berendezés végre tud hajtani. Az NS(PP) halmaz a gyökér részfeladatnál (e, N_e) párokat tartalmaz, ahol $e \in E$ és $N_e \subset N$ a recept-gráf alapján az e berendezéssel végrehajtható taszk-csúcsok halmaza.

3.4.2. Szétválasztás eljárás

Az EQ-SG algoritmus szétválasztási eljárása felelős azért, hogy a keresési tér hatékony és szisztematikus bejárásával megkapjuk az optimális ütemezési-gráfot. A Sanmartí és társai által bemutatott S-gráf alapalgoritmus az úgynevezett „berendezés alapú döntéseket” használja [86]. Az EQ-SG algoritmus minden vizsgált részfeladatnál kiválaszt

procedure *EQ-SG*
jelölések:
N_e: az e berendezéssel végrehajtható taszkok halmaza ($e \in E$)
$PP = (G(N, A_1, A_2), bound, last\_node, SOUN, NS)$: részfeladat
G(PP): a részfeladathoz tartozó S-gráf
bound(PP): a részfeladathoz tartozó *bound* érték
last_node(PP)=$\{(e, j) : e \in E$ és $j \in N_t\}$: a részfeladathoz tartozó *last_node* halmaz, melynek elemei a berendezések utolsónak ütemezett taszkjait jelzik
SOUN(PP): a részfeladathoz tartozó $SOUN$ halmaz
NS(PP)=$\{(e, H) : e \in E\}$: a PP részfeladathoz tartozó NS halmaz, melynek elemei a berendezésekkel végrehajtható taszkokat jelzik
SET: nyitott részfeladatokat tartalmazó halmaz
bemenet:
$G(N, A_1, \emptyset)$ recept-gráf
begin
N_e halmazok meghatározása ($e \in E$);
$SET := \emptyset$;
$current\_best := \infty$;
G(PP):= $G(N, A_1, \emptyset)$;
last_node(PP):= $\emptyset$;
SOUN(PP):= N_t;
NS(PP):= $\{(e, N_e) : e \in E\}$;
bound(PP):=*EQ-korlátozás*(PP);
if bound(PP)$<\infty$
begin
$SET := SET \cup \{PP\}$;
while $SET \neq \emptyset$
begin
vegyünk ki egy elemet SET-ből, jelöljük PP-vel;
EQ-szétválasztás(PP);
end
end
end.
kimenet:
current_best tartalmazza az optimum értékét
solution tartalmazza az optimális ütemezési-gráfot

3.5. ábra. Az EQ-SG algoritmus.

egy berendezést ütemezésre, majd hozzárendel a berendezés utoljára végrehajtandó taszkjának a lehetséges végrehajtható taszkok közül egyet az összes számba vehető módon. Az értekezés 4. fejezetében bemutatom az S-gráf módszertanba integrált új szétválasztási eljárást, mely a „taszk alapú döntéseket" használja.

A szétválasztás és korlátozás algoritmus futása során a vizsgált részfeladatokhoz tartozó részleges ütemezések S-gráfokkal írhatóak le. Szétválasztási lépések során S-gráfok sorozatán halad az EQ-SG algoritmus a recept-gráfból (gyökér szint) az ütemezési-gráf irányába (levél szint). Bármely szülő részfeladathoz tartozó S-gráfra és a belőle származtatott gyermek részfeladat S-gráfjára teljesül, hogy a gyermek részfeladat S-gráfjának része a szülő részfeladat S-gráfja oly módon, hogy a gráfok csúcsai megegyeznek, a szülő részfeladat S-gráfjának minden éle szerepel a gyermekhez tartozó S-gráfban, legfeljebb a recept-élek értékében térhet el. A szétválasztási lépések során a szülő részfeladat S-gráfjának a csúcsai és élei nem változnak, legfeljebb a recept-élek értékei növekedhetnek, illetve új ütemezési-élekkel bővülhet a gyermek részfeladat S-gráfja.

A 3.6 ábra az *EQ-szétválasztás* eljárásának a pszeudó kódját tartalmazza. Egy szétválasztási lépés során az aktuális részfeladatnál kiválasztunk egy olyan berendezést (EQ változó), mely végrehajthat még olyan taszkot, mely taszk nincsen ütemezve. Az SO halmaz tartalmazza az EQ berendezéshez rendelhető nem ütemezett taszkokat. Az SX halmaz azokat az SO-beli taszkokat tartalmazza, melyeket az EQ berendezésen kívül valamely másik berendezés is végre tud még hajtani. A *szétválasztás* eljárás egy iterációs lépés során az összes SO halmazbeli lehetőséget végignézve hozzárendeli a következően végrehajtandó taszkokat a berendezéshez és létrehozza a megfelelő gyermek részfeladatokat.

Először a részfeladatból származtatjuk az összes lehetséges gyermek részfeladatot ($CHILD$ változó) a be nem ütemezett, végrehajtható taszkok alapján. A gyermek részfeladatokban a berendezést hozzárendeljük egy megfelelő, be nem ütemezett taszkhoz, és a taszkot beütemezzük a berendezés aktuálisan utolsónak végrehajtandó taszkjának. Mivel a taszkok működési ideje függhet a hozzá rendelt berendezéstől, ezért hozzárendelés esetén módosulhat a taszkhoz tartozó recept-él súlya. A last_node(PP) halmaz alapján ütemezési-éllel jelöljük a berendezés ütemezését a

részfeladat S-gráfjában. Ha az új részfeladat megvalósítható és alsó korlátja kisebb, mint az eddigi legjobb megoldás, akkor a gyermek részfeladatot berakjuk a nyitott részfeladatok halmazába.

A keresési tér teljes bejárásához, azon taszkoknál, melyek az EQ berendezésen kívül legalább egy másik berendezéssel is végrehajthatóak, létre kell hoznunk egy olyan $CHILD$ részfeladatot, melyben az EQ berendezés már nem végezhet egyetlen egy új taszkot sem. Azért, hogy ugyanazt a részfeladatot ne generáljuk többször a keresési fában, csak akkor zárjuk ki az EQ berendezést további taszkok végrehajtásából, ha már ütemeztük az összes olyan taszkot, melyet csak az EQ berendezéssel lehet végrehajtani azaz, hogyha teljesül, hogy az $SX = SO$. Ekkor az SX halmazbeli taszkok végrehajtásából kizárjuk az EQ berendezést, az SX halmazbeli taszkok végrehajtási idejét módosítjuk annak a berendezésnek a működési idejére, mely a taszkot aktuálisan végrehajthatja a taszkot és a legrövidebb.

3.4.3. Korlátozás eljárás

A korlátozás eljárás a vizsgált részfeladat ütemezésének megvalósíthatósági vizsgálatából és az ütemezés végrehajtási idejének alsó korlát számításából áll. Azok az ütemezések megvalósíthatóak, melyekhez tartozó S-gráfok nem tartalmaznak irányított kört. A számított alsó korlát a részfeladatból származtatott összes megvalósítható ütemezés végrehajtási idejére alsó korlát. A részfeladatok alsó korlátjának meghatározására az EQ-SG algoritmus a leghosszabb út kereső algoritmust használja. Az *EQ-korlátozás* eljárás pszeudó kódja a 3.7 ábrán látható.

Egy részfeladathoz tartozó S-gráf mindig részgráfja bármely belőle származtatott részfeladat S-gráfjának. Mivel az *EQ-szétválasztás* eljárásban legfeljebb új, nemnegatív súlyú ütemezési-élek hozzáadása és a recept-élek súlyának növelése történhet, ezért a szülő részfeladatból származtatott gyermek részfeladatok S-gráfjának leghosszabb útja nem csökkenhet a szülő részfeladat alsó korlátjához képest.

Ha egy részfeladat S-gráfja irányított kört tartalmaz, akkor bármely belőle származó összes gyermek részfeladat tartalmazni fogja ugyanezt a kört. A kört tartalmazó S-gráfok nem megvalósítható ütemezéseket jelentenek, ezért a kört tartalmazó S-gráf

```
procedure EQ-szétválasztás(PP)
jelölések:
    NS_e(PP) = N*, ahol (e, N*) ∈ NS(PP) és e ∈ E: a PP részfeladatban
        az e berendezéssel végrehajtható taszkok halmaza
begin
    if BOUND(PP)<current_best
        begin
            legyen EQ egy olyan berendezés, melyhez NS_EQ(PP)∩SOUN(PP) ≠ ∅;
            SO := NS_EQ(PP) ∩ SOUN(PP);
            SX := {t : t ∈ SO és ∃e ∈ E, e ≠ EQ, ahol t ∈ NS_e(PP)};
            for all k ∈ SO
                begin
                    CHILD := PP;
                    for all (k,l) ∈ A_1, ahol G(CHILD)=(N, A_1, A_2)
                            c(k,l) súly módosítása G(CHILD) S-gráfban;
                    if ∃(EQ,i) ∈ last_node(CHILD)
                            for all (i,j) ∈ A_1, ahol G(CHILD)=(N, A_1, A_2)
                                    G(CHILD):= (N, A_1, A_2 ∪ {(j,k)});
                    SOUN(CHILD) :=SOUN(CHILD)\k;
                    if ∃(EQ,i) ∈ last_node(CHILD)
                            last_node(CHILD):=last_node(CHILD)∪{(EQ,k)} \ {(EQ,i)};
                    else
                            last_node(CHILD):=last_node(CHILD)∪{(EQ,k)};
                    for all e ∈ E
                            if k ∈ NS_e(CHILD)
                                    NS_e(CHILD) := NS_e(CHILD) \ {k};
                    bound(CHILD):=EQ-korlátozás(CHILD);
                    if bound(CHILD)<current_best
                            if SOUN(CHILD)=∅
                                    current_best és solution módosítása;
                            else
                                    SET := SET ∪ {CHILD};
                end
            if SX = SO
                begin
                    CHILD := PP;
                    NS_EQ(CHILD) := NS_EQ(CHILD) \ SX;
                    for all (k,l) ∈ A_1, ahol G(CHILD)=(N, A_1, A_2) és k ∈ SX
                            c(k,l) súly módosítása G(CHILD) S-gráfban;
                    bound(CHILD):=EQ-korlátozás(CHILD);
                    if bound(CHILD)<current_best
                            SET := SET ∪ {CHILD};
                end
        end
end.
```

3.6. ábra. Az EQ-SG algoritmus *EQ-szétválasztás* eljárása.

```
procedure EQ-korlátozás(PP)
begin
    kör_kereső(G(PP));
    if no cycle then
        bound :=leghosszabb_út_kereső(G(PP));
    else
        bound := ∞;
    return bound;
end.
```

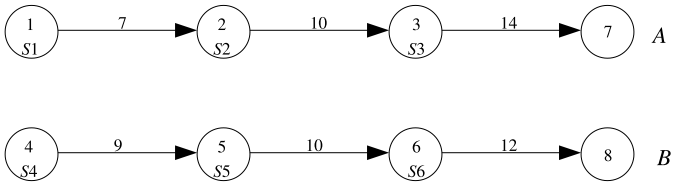
3.7. ábra. Az EQ-SG algoritmus *EQ-korlátozás* eljárása.

részfeladatát nem kell tovább vizsgálni, nem lehet a részfeladatból megvalósítható ütemezést kapni.

A leghosszabb út kereső algoritmus a keresési fa gyökerének közelében nem szolgáltat elég éles alsó korlátot, mivel az S-gráfok ekkor még nem eléggé összefüggőek, hiszen kevés ütemezési-él szerepel a gráfban. Az alsó korlát a leghosszabb út kereső algoritmus helyett bizonyos részfeladatoknál lényegesen jobban becsülhető lineáris programozási modellek segítségével [34, 33]. A körkereső és a leghosszabb út kereső algoritmus pszeudó kódja a B és C függelékekben található.

3.5. Az EQ-SG algoritmus működésének szemléltetése

Két NIS típusú szakaszos ütemezési feladat megoldásán keresztül szemléltetem az EQ-SG algoritmus működését, lépéseit. A 3.1. feladatban mindegyik taszk pontosan egy berendezéssel végezhető el a recept alapján, a 3.2. feladat általánosabb ütemezési feladat taszkokhoz rendelhető alternatív berendezésekkel.



3.8. ábra. A 3.1. feladat recept-gráfja.

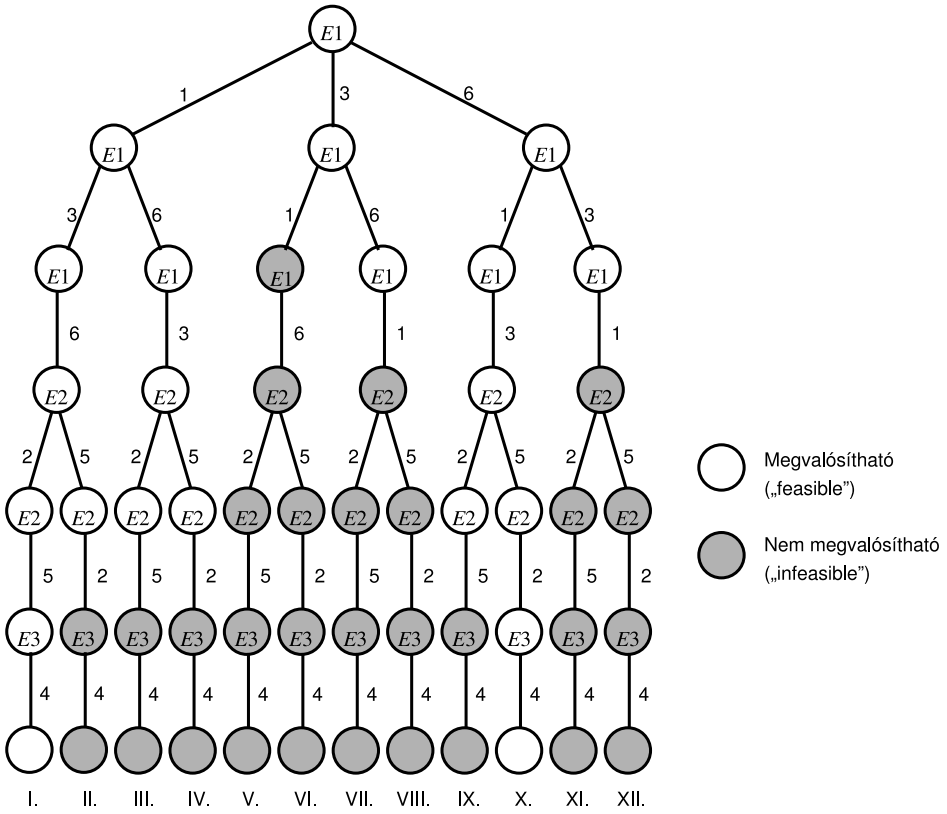
3.1. feladat

Tekintsük a 3.8 ábrán található A és B terméket előállító recept-gráfot. Mindkét termék három egymás utáni taszkkal állítható elő. Legyen az $S1 = \{E1\}$, $S2 = \{E2\}$, $S3 = \{E1\}$, $S4 = \{E3\}$, $S5 = \{E2\}$ és $S6 = \{E1\}$.

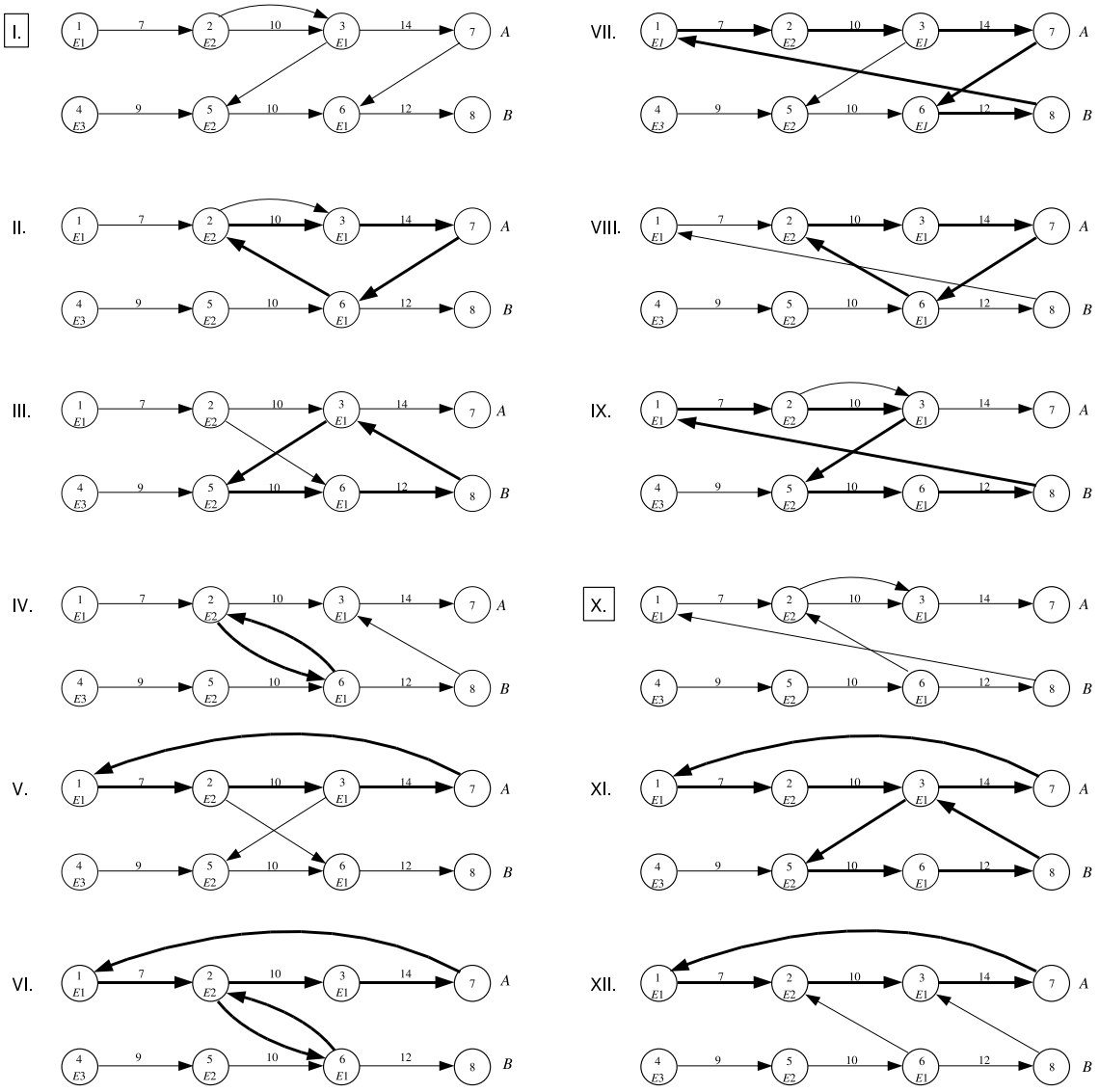
A keresési tér a hozzá tartozó keresési fával ábrázolható, mely fa bejárása függ a szétválasztási eljárásban hozott döntések sorrendjétől. Ha az *EQ-szétválasztás* eljárásban a berendezéseket az $E1, E2, E3$ sorrendben választjuk ki ütemezésre, akkor a 3.9 ábra keresési fáját vizsgáljuk. A keresési fa csúcsaiban a berendezés azonosítója szerepel, melyre az algoritmus a szétválasztás lépésben a döntést hozza. A keresési fa élei a lehetséges döntéseket jelzik, azaz annak a taszknak az azonosítóját, mely utolsónak beütemezésre kerül a a döntésre kiválasztott berendezéshez. Ennek a keresési fának egy részét járja be az EQ-SG algoritmus.

A keresési fa levelei S-gráfokat jelölnek, mely S-gráfok között vannak a feladat lehetséges megoldásai. A lehetséges megoldások speciális S-gráffal, az ütemezési-gráffal adhatóak meg. A keresési fa gyökere és egy levele közötti élek címkéinek sorrendje azonosítja a feladat egy ütemezését, azaz a berendezésekhez rendelt végrehajtandó taszkokat és azok sorrendjét. A tizenkét levél közül 2 darab jelöl megvalósítható ütemezést (az ábra I. és X. jelű levele), és 10 darab nem megvalósítható ütemezést reprezentál. A 3.10 ábrán a teljes leszámolási keresési fa leveleihez tartozó megoldások S-gráfjait láthatjuk. A nem megvalósítható ütemezéseket, melyek irányított kört tartalmaznak, a keresési fában szürke kitöltésű levelek jelölik.

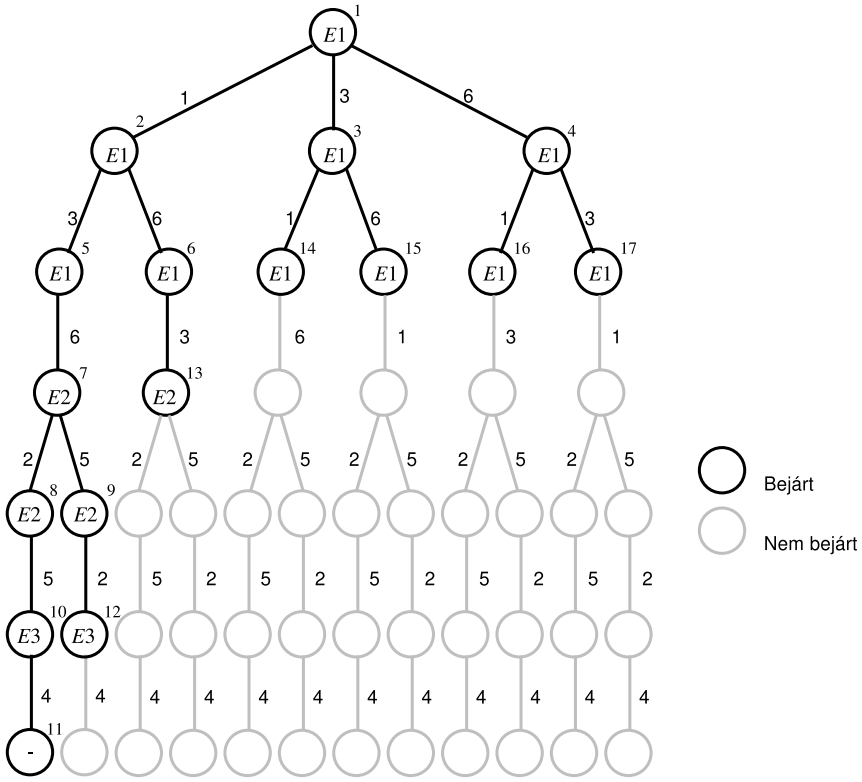
Az implementált EQ-SG algoritmus nem járja be a teljes keresési fát a feladat megoldása során, a részfeladatok megvalósíthatóságát és a végrehajtási idő alsó korlátját



3.9. ábra. A 3.1. feladat teljes leszámolási fája (korlátozási lépés nélkül).



3.10. ábra. A 3.1. feladat teljes leszámolási fájának leveleihez tartozó S-gráfok. Körök vastagított élekkel jelölve.



3.11. ábra. A 3.1. feladat megoldása során bejárt keresési fa a korlátozás lépés alapján. (Az optimális megoldás a 11-es részfeladat, a végrehajtási ideje: 43.)

vizsgálva a keresési tér jelentős részét nem kell bejárnia. A 3.11 ábra az EQ-SG ütemező algoritmus során bejárt keresési fát ábrázolja. A megvalósításnál az algoritmus a keresési fát mélységi keresés alapján járja be, a szétválasztás eljárás a berendezéseket az $E1, E2, E3$ sorrendben ütemezi. A korlátozási eljárás során a részfeladat S-gráfjának körmentességét vizsgáltam, alsó korlát számításra pedig a leghosszabb út kereső algoritmust alkalmaztam. A részfeladatok generálásának és megvizsgálásának sorrendjét a keresési fa csúcsainak számozása mutatja. A részfeladatok korlátozás eljárásban számolt alsó korlátjait a 3.1 táblázatban találhatjuk.

3.1. táblázat. A 3.1. feladat megoldása során vizsgált részfeladatokra számított alsó korlát értékek (azonosítók a 3.11 ábrához kapcsolódnak).

Azonosító	Alsó korlát	Azonosító	Alsó korlát	Azonosító	Alsó korlát
1	31	7	43	13	45
2	31	8	43	14	∞^*
3	31	9	43	15	43
4	31	10	43	16	62
5	31	11	43	17	45
6	31	12	∞^*		

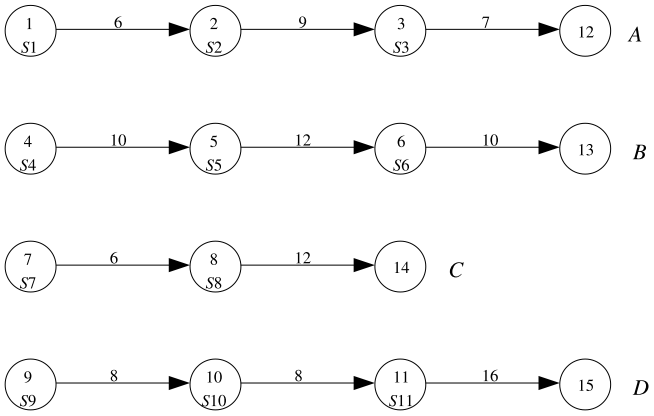
* nem megvalósítható

3.2. táblázat. A 3.2. feladat receptje.

Taszk	A termék		B termék		C termék		D termék	
	Beren-dezés	Idő (h)	Beren-dezés	Idő (h)	Beren-dezés	Idő (h)	Beren-dezés	Idő (h)
1	$E1$	6	$E4$	10	$E2$	6	$E3$	8
	$E2$	8			$E3$	10		
2	$E3$	9	$E3$	15	$E4$	12	$E1$	9
			$E4$	12	$E5$	16	$E3$	8
3	$E2$	7	$E1$	10			$E4$	18
	$E5$	7	$E2$	17			$E5$	16
			$E3$	13				

3.2. feladat

A 3.2. feladatban négy termékből, az A, B, C és D termékből kell egy-egy batchnyi mennyiséget optimálisan ütemezni. A termékek előállítására az $E1$, $E2$, $E3$, $E4$ és $E5$ berendezéseket használhatjuk. A termékek gyártását leíró recept a 3.2 táblázatban található. A 3.2. feladat recept-gráfja a 3.12 ábrán található. Az ábrán a taszkokhoz tartozó Si halmazok a következőek: $S1 = \{E1, E2\}$, $S2 = \{E3\}$, $S3 = \{E2, E5\}$, $S4 = \{E4\}$, $S5 = \{E3, E4\}$, $S6 = \{E1, E2, E3\}$, $S7 = \{E2, E3\}$, $S8 = \{E4, E5\}$, $S9 = \{E3\}$, $S10 = \{E1, E3\}$ és $S11 = \{E4, E5\}$.

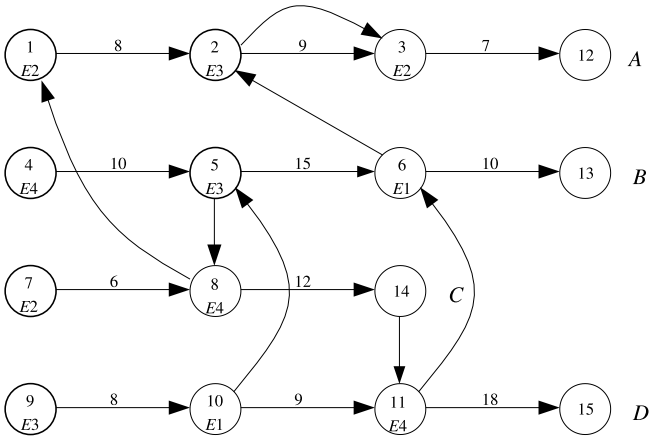


3.12. ábra. A 3.2. feladat recept-gráfja.

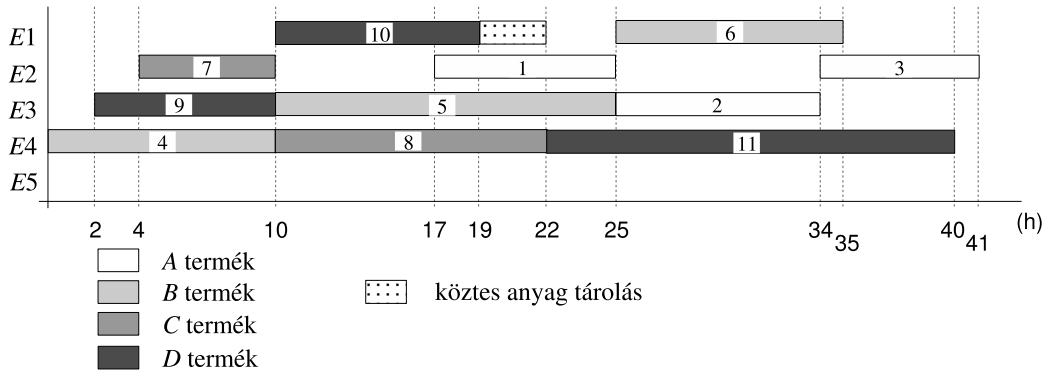
A 3.2. feladat minimális végrehajtási idejű ütemezése 41 óra időtartamú. Az EQ-SG algoritmus a 3.13 ábrán látható ütemezési-gráfot adta. Ez egy olyan optimális megoldása a feladatnak, melyben az $E5$ berendezést nem kell használni. A 3.14 ábra a megoldás Gantt-diagrammját ábrázolja.

3.6. Az S-gráf alapalgoritmus keresési terének csökkentése: egy termékből több batch előállítása

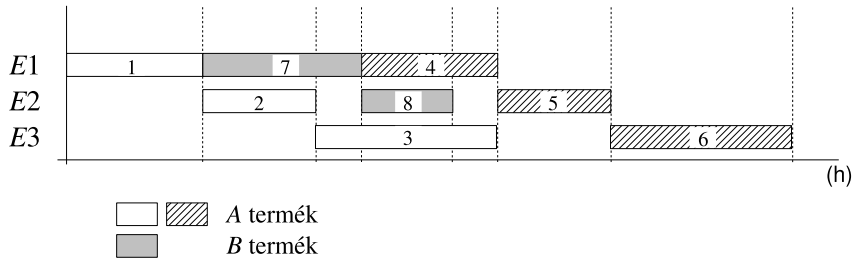
Az S-gráf módszertan alapalgoritmusa azokat a feladatokat, melyekben egy termékből több batchnyit kell előállítani és ütemezni nem kezeli kellően hatékonyan [86]. Az algoritmus ugyanis a megoldások keresése közben külön termékként kezeli az ugyanahhoz a termékhez tartozó többi batchet. Ugyanakkor a kapott megoldásokat elemezve észrevehetjük, hogy gyakorlatilag ugyanazokat az optimális megoldásokat kapjuk meg többször (technikailag ugyanaz az ütemezés). Természetesen elég egy optimális ütemezést megkapnunk. A keresési tér csökkentésével a technikailag azonos ütemezések kiszűrhetőek és így jelentős gyorsítást érhetünk el a megoldás menetében, miközben a megoldás optimalitása továbbra is megmarad.



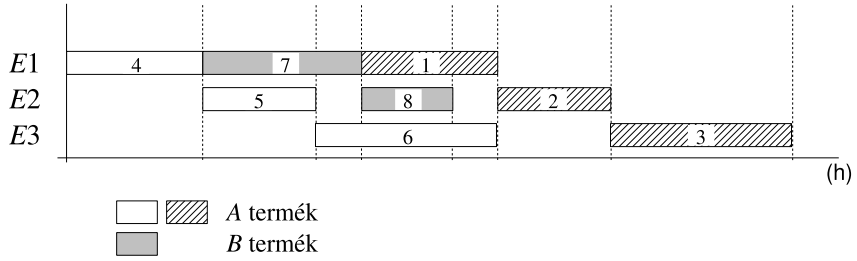
3.13. ábra. A 3.2. feladat egy optimális ütemezési-gráfja.



3.14. ábra. A 3.2. feladat 3.13. ábrán látható ütemezésének Gantt-diagrammja.



3.15. ábra. Ütemezés Gantt-diagrammja.

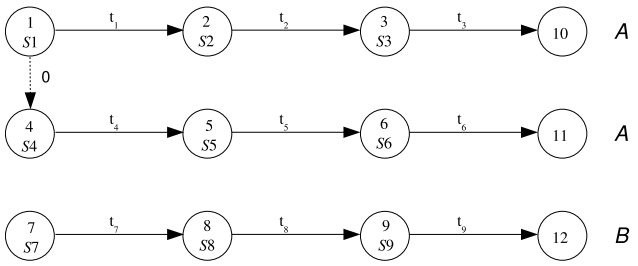


3.16. ábra. A 3.15 ábra ütemezésével technikailag azonos ütemezés.

3.6.1. Technikailag ekvivalens ütemezések

Egy egyszerű példa segítségével mutatom be, hogy miért nem lehet hatékony, ha azonos termék batcheit megkülönböztetve, külön termékként kezelve ütemezzük, ahelyett, hogy kihasználnánk azt, hogy a két batch taszkjai között nincs különbség. Tegyük fel, hogy két batchnyi A és egy batchnyi B terméket kell előállítani. A 3.15 ábra Gantt-diagrammja a feladat egy ütemezését ábrázolja. Ebből az ütemezésből könnyen kaphatunk egy másik ütemezést, hogyha felcseréljük az A termék két batchéhez tartozó megfelelő taszkokat. A 3.16 ábra az előző ütemezéssel technikailag ekvivalens ütemezést ábrázolja. Mindkét ütemezés végrehajtási ideje ugyanolyan értékű, az ütemezések csak elméletileg térnek el egymástól.

A technikailag ekvivalens ütemezések között a gyakorlatban, a gyártási folyamatban nincsen különbség. Két technikailag azonos ütemezés bármely berendezése ugyanazokat a taszkokat (gyártási műveleteket) hajtja végre ugyanabban a sorrendben. Így



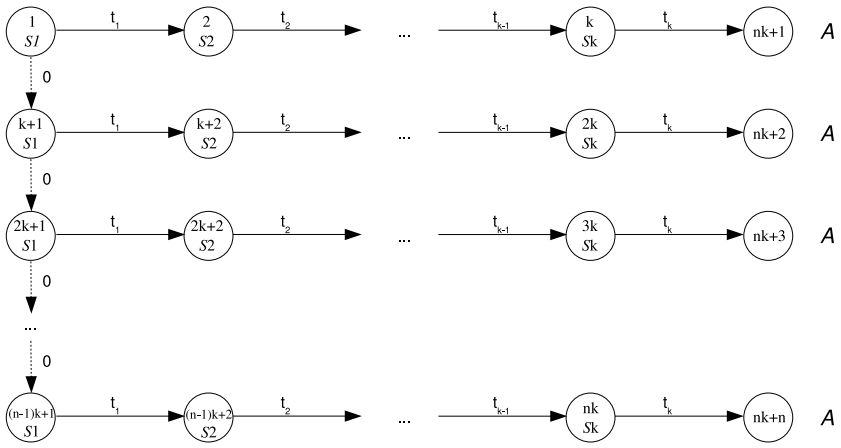
3.17. ábra. A 3.16 ábra ütemezését kizáró recept-gráf.

a különbözőség, csak az elméletben létezik. Az ütemezések között a megkülönböztetés abból ered, hogy az algoritmus és a megfelelő ábrázolási technikák céljából sorszámoztuk és azonosítottuk a különböző batchekhez tartozó ugyanazon taszkokat.

Ahogy a példa is mutatja az EQ-SG algoritmus önmagában nem tudja kizárni a keresési térből a technikailag azonos, ekvivalens ütemezéseket. Az algoritmus szétválasztás és korlátozás jellege miatt az optimális ütemezés összes technikailag azonos ütemezése generálásra kerülhet, mert a korlátozás eljárás alapján a technikailag ekvivalens ütemezésekhez vezető részproblémák nagy része nem zárható ki. Ugyanakkor nagy méretű ipari ütemezési feladatokban nagy mennyiségben kell egyazon termék több batchét ütemezni. A nagy számú azonos termékhez tartozó ütemezendő batch és az EQ-SG algoritmus jellege miatt ezeket a feladatokat nehéz megoldani. A keresési térből kizárhatjuk a technikailag azonos ütemezéseket, ha bevezetünk néhány feltételt a receptbe. Ha feltesszük, hogy a 4-es taszk nem kezdődhet korábban, mint az 1-es taszk, akkor a 3.16 ábra ütemezését az S-gráf algoritmus használatával már nem kapjuk meg. Ezt a feltételt a recept-gráfban egy nulla súlyú „segéd-él" (*auxiliary-arc*) behúzásával jelölhetjük, mely él az 1-es taszkból a 4-es taszkba mutat (lásd. 3.17 ábra). Ez az él nem tartozik se a recept-gráfhoz, se az ütemezési-gráfhoz, ezért az ábrákon szaggatott vonallal fogom jelölni.

3.6.2. Recept-gráf kiegészítése segéd-élekkel

Ha egy termékből n batchnyit kell előállítani, akkor a technikailag azonos ütemezések kizárása nélkül $n!$ számú technikailag azonos megoldást kaphatunk a termék

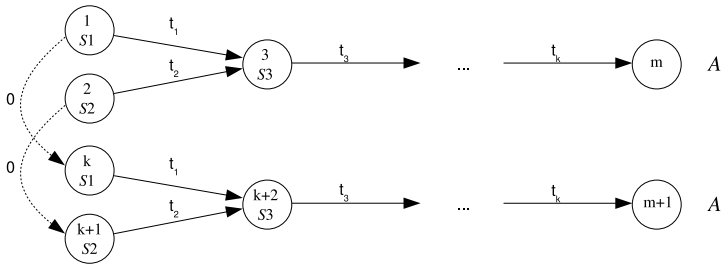


3.18. ábra. Egyszerű recept recept-gráfja segéd-élekkel kiegészítve, ha n batchnyit kell egy termékből ütemezni.

ütemezése során. Ezek a megoldások csak a termékhez tartozó batchek sorrendjében különböznek. Az n növekedésével a lehetséges permutációk száma faktoriálisan növekszik. A batchekhez tartozó első taszkok sorrendbe rendezésével – például az első batch első taszkja nem kezdődhet később, mint a második batch első taszkja – a batchek felesleges permutációját ki tudjuk zárni a keresési térből. A redundancia kizárását a recept-gráfban a batchek rendezésével, azaz nulla súlyú segéd-élek bevezetésével tudjuk ábrázolni. A 3.18 ábrán n batchnyi termék receptje látható kiegészítve segéd-élekkel. A segéd-élek bevezetése nem zárja ki az optimális megoldás megtalálását, mivel minden optimális megoldáshoz létezik $n! - 1$ darab technikailag azonos szintén optimális ütemezés.

Összetett recept esetén esetén, ha egy termék gyártásának több olyan tevékenysége van, mely tevékenységnek nincsen megelőző tevékenysége, akkor a redundancia kizárható a megfelelő kezdő csomópontok segéd-élekkel való összekötésével. A 3.19 ábrán összetett, több első tevékenységet tartalmazó recept két batchnyi terméket tartalmazó recept-gráfját találhatjuk segéd-élekkel kiegészítve.

Holczinger Tibor disszertációjában [33] igazolta, hogy tetszőleges ütemezési-gráfot ki lehet egészíteni az egyazon termékhez tartozó batchek első taszk-csúcsai közötti segéd-élekkel oly módon, hogy a segéd-élek nem okoznak kört és nem növelik meg az



3.19. ábra. Összetett recept recept-gráfja segéd-élekkel kiegészítve.

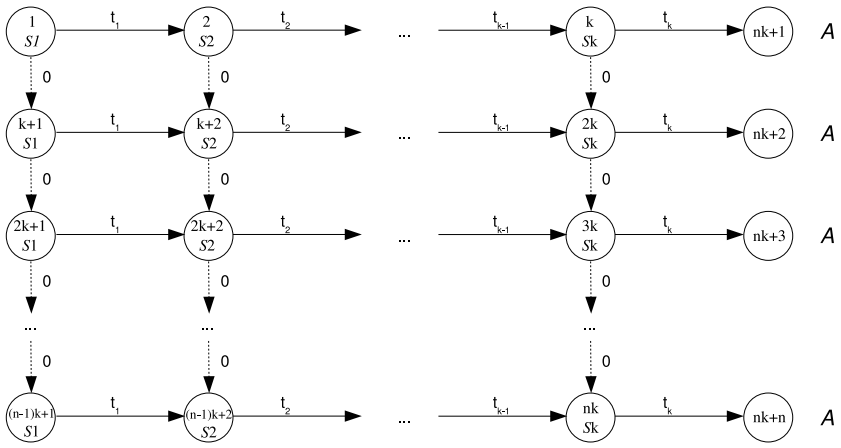
S-gráf leghosszabb útjának hosszát.

3.6.3. Redundanciát kizáró feltételek további élesítése

Bizonyos receptek esetén az előzőleg bemutatott segéd-élek tovább finomíthatóak, tovább élesíthető az alsó korlát. Az előbb leírt módszer általános, azonban tovább élesítve a feltételeket az S-gráf módszertan alapalgoritmusának hatékonysága javul.

Ilyen eset, amikor minden taszk elvégzéséhez pontosan egy berendezés áll a rendelkezésre a recept alapján. Ebben az esetben nem csak az azonos termékhez tartozó batchek első csomópontjai között állíthatunk fel egy rendezést a segéd-élek behúzásával a recept-gráfba, hanem a batchek összes többi csomópontja között megvalósíthatjuk ugyanazt a rendezést további segéd-élek bevezetésével az elsővel azonos irányban. A 3.20 ábra a segéd-élekkel kiegészített recept-gráfot szemlélteti, ha minden taszk csak egy berendezéssel végezhető el a recept alapján.

Ha NIS tárolási stratégiát alkalmazunk, a feltételeket tovább lehet élesíteni. Ha minden taszkhoz pontosan egy berendezés tartozik, akkor a segéd-él kezdőpontja a recept szerinti következő tevékenységet jelölő S-gráf csomópont lehet. A segéd-élek minél jobban élesítik a részfeladatok alsó korlátját az S-gráf módszertan alapalgoritmusában, annál jobban gyorsítják az algoritmust az optimum keresésben [34].



3.20. ábra. Recept-gráf segéd-élekkel kiegészítve, ha minden taszk csak egy berendezéssel hajtható végre.

3.6.4. Szemléltető feladat a segéd-élek alkalmazására

A segéd-élek hatékonyságát mutatom be az S-gráf alapalgoritmus futási eredményével illusztrálva. Egy NIS ütemezési feladat segéd-élek nélkül és segéd-élekkel együtt megoldva hasonlítom össze az algoritmust.

3.3. feladat

A 3.2. feladatban bemutatott gyártási folyamat receptjét kell beütemezni a 3.3. feladatban az EQ-SG algoritmussal az eredeti és a segéd-éleket tartalmazó recept-gráfokra. A recept alapján az A és B termékből három-három batchnyit, a C és D termékből egy-egy batchnyit kell beütemezni a rendelkezésre álló $E1$, $E2$, ..., $E5$ berendezéseket használva.

Az ütemezési feladatot a segéd-éleket nem tartalmazó recept-gráfból kiindulva megoldva az optimális ütemezés végrehajtási ideje 87 óra lett. A feladat megoldásához szükséges idő 79,96 másodperc volt[1].

Ha ugyanezt a feladatot a segéd-éleket tartalmazó recept-gráfból kiindulva oldjuk meg ugyanazzal az EQ-SG algoritmussal, ugyanazon a számítógépen, a megoldáshoz

[1]Az algoritmus egy 1,5 GByte DDR RAM-mal felszerelt, Intel Pentium 4 3.06GHz órajelű PC-n futott.

szükséges idő 37,86 másodperc volt. A kapott ütemezés végrehajtási ideje szintén 87 óra, azonban az ütemezések különbözőek. A feladat méretének növelésével, azaz az ütemezendő batchek számának növelésével a segéd-éleket tartalmazó recept-gráf ütemezése még hatékonyabb a segéd-él nélkülihez képest.

4. fejezet

Taszk alapú döntési stratégia ütemezési feladatok megoldása során

Az előző fejezetben bemutattam az S-gráf módszertant és a módszertan alapján implementált EQ-SG algoritmust. Sanmartí és társai az S-gráf módszertant a szakaszos folyamatok ütemezésére vezették be [86]. A módszertant eredetileg az NIS tárolási stratégia megoldására fejlesztették ki, a kombinatorikus gyorsítások és az S-gráf ábrázolás az NIS stratégiát használó ütemezési feladatosztály speciális tulajdonságait használják ki. Az S-gráf módszertan tartalmazza a feladat és a megoldások megfelelő ábrázolását [84], a megoldó alapalgoritmust és a lehetőségeket illetve ajánlásokat további gyorsítások eléréséhez. Az értekezésnek ebben a fejezetében az S-gráf módszertan *EQ-szétválasztás* eljárására mutatok be egy új eljárást, mely a keresési tér újfajta bejárásával oldja meg az ütemezési feladatokat.

Szakaszos ütemezési feladatok tulajdonságainak elemzése alapján jelentős gyorsítások érhetőek el a feladatosztály kombinatorikus tulajdonságainak kihasználásával az eredeti EQ-SG algoritmushoz képest. Az EQ-SG algoritmus kombinatorikus tulajdonságok alapján történő gyorsítása helyett a új elveken felépített ütemező algoritmussal bizonyos feladatokra hatékonyabb ütemezőt kaphatunk. Ebben a fejezetben az S-gráf módszertanban bemutatott szétválasztás és korlátozás algoritmust új ötletek alapjain építem fel. Egy olyan új szétválasztási eljárást mutatok be, mely ugyanabban a keresési térben biztosítja az ütemezési feladat optimális megoldásának megtalálását.

Az EQ-SG algoritmus a szétválasztási eljárásában a „berendezés alapú döntési stratégiát” használja. A döntési stratégia során minden döntésnél a részfeladat egy ütemezendő berendezésének a kiválasztása után hozzáfűzünk a berendezés végrehajtási sorának végére egy még be nem ütemezett taszkot. A gyermek részfeladatok az összes lehetséges taszk hozzáfűzést figyelembe véve keletkeznek. Ennek a fejezetnek a témáját az új, úgynevezett „taszk alapú szétválasztási stratégia” képezi. Az EQ-SG algoritmus szétválasztási eljárásában a berendezéshez rendeljük a taszkot, az új szétválasztási eljárásban viszont a taszkhoz rendeljük hozzá a berendezést. A végeredmény szempontjából mindegy a kiválasztott és a hozzárendelt attribútum, azonban ennek fontos szerepe lehet az algoritmus hatékonyságának szempontjából. A „taszk alapú döntéseket” használó szétválasztás eljárásban a döntések új alapötlete az, hogy rendeljünk hozzá egy még be nem ütemezett taszkhoz egy lehetséges berendezést és szúrjuk be a taszkot a berendezés végrehajtási sorrendjébe figyelembe véve az összes lehetőséget.

4.1. Berendezés alapú döntési stratégia előnyei és hátrányai

A „berendezés alapú döntési stratégiát” használó EQ-SG algoritmus [84, 86] alapja, hogy a döntéseink során próbáljuk meg meghatározni minden egyes berendezés ütemezését oly módon, hogy rendeljük az aktuális ütemezésre kiválasztott berendezéshez azt az új taszkot, amit utoljára fog végrehajtani. Két fajta taszk lehet egy részfeladatban: vannak olyan taszkok, melyek már hozzá vannak rendelve valamelyik berendezéshez és ütemezve vannak, és vannak olyanok, melyek még nincsenek ütemezve. A szétválasztási eljárás során, hogy minden egyes berendezéshez az eddig hozzárendelt és beütemezett taszkok rögzítettek, ezek sorrendjén már nem változtatunk. A döntési alternatívák a berendezés aktuális, utoljára végrehajtandó taszkjának kiválasztásánál vannak. Azért, hogy a keresési tér összes ütemezését megkapjuk, figyelembe kell venni azt a döntési lehetőséget is, hogy az aktuális berendezés nem fogja végrehajtani azokat a taszkokat, amik végrehajthatók valamelyik másik berendezéssel is.

Az EQ-SG algoritmus minden részfeladata egy részütemezést jelképez, mely részütemezés leírható egy S-gráf formájában. Az S-gráf ütemezett és ütemezésre váró taszkokat tartalmaz, minden berendezéshez adott a hozzá rendelt végrehajtandó taszkok sora. Ha az S-gráf alapalgoritmust olyan feladatosztály megoldására adoptáljuk, mely megköveteli a recept dinamikus változtatását (például a recept-gráf ütemezése során eldől, hogy valamelyik termékből további batcheket kell előállítani, és a hozzá tartozó újonnan felvett taszkokat ütemezni kell), akkor a részfeladat részlegesen beütemezett S-gráfjából kiindulva nem lehet az ütemezést úgy folytatni, hogy az optimális megoldást garantáltan megkapjuk. Ilyen feladatokat kapunk on-line jellegű ütemezési feladatokat vizsgálva. Ilyen esetekben nem lehet hatékonyan használni a berendezés alapú döntési stratégiát használó EQ-SG algoritmust.

Ha az ütemezési feladatban szerepelnek olyan berendezések, melyek sok taszkot végre tudnak hajtani, akkor ezeknek a berendezéseknek az ütemezése nagy számú döntési lehetőséggel jár. A nagy számú döntési lehetőség eredményeként széles keresési fát kapunk, mely bejárása rendkívül nagy számítási igénnyel járhat.

Az EQ-SG algoritmus hatékonyan megtalálja a feladat optimális ütemezését, ha talál a receptben egy olyan berendezés halmazt, melyben levő berendezések ütemezése döntően meghatározzák a megoldás végrehajtási idejét. Az ilyen berendezés halmazt a feladat szűk keresztmetszetének (*bottleneck*) is szokás nevezni. Ennek a berendezés halmaznak az ütemezése után a többi berendezés ütemezése már nem befolyásolja az ütemezés végrehajtási idejét, így a keresési tér jelentős részét nem kell megvizsgálni.

Más jellegű szétválasztási eljárással hatékonyabb algoritmusokat kaphatunk. Ezek a hátrányos tulajdonságok is indokolják új döntési stratégiák bevezetését.

4.2. Algoritmus a taszk alapú döntési stratégia alapján

Ebben a fejezetben a „taszk alapú döntési stratégiát" megvalósító S-gráf alapú ütemező algoritmus (továbbiakban TA-SG algoritmus) működését mutatom be. A TA-SG algoritmus egy szétválasztás és korlátozás elvén működő algoritmus. Az algoritmus működésének ismertetéseként bemutatom az algoritmus futása során kezelt részfeladatokat, a nyitott részfeladatokból új részfeladatokat származtató szétválasztás eljárást és a részfeladatok megoldhatóságát és a belőlük kapható megoldások alsó korlátját számító korlátozás eljárást.

4.2.1. Részfeladat ábrázolása és az adatstruktúrák inicializálása

A TA-SG algoritmus, mint minden szétválasztás és korlátozás elvén működő algoritmus, egy keresési fát valamilyen módon bejárva határozza meg a feladat optimális megoldását. A keresési fa csúcspontjaihoz tartoznak a részfeladatok. Minden részfeladatnak tárolnia kell a keresési fa gyökeréből indulva a részfeladathoz vezető éleken meghozott döntéseket.

A PP részfeladatot definiáljuk a $(G(N, A_1, A_2), bound, EQLIST, SOUN)$ négyessel. A részfeladathoz tartozó részütemezés egy S-gráf segítségével megadható. Jelölje G(PP) a PP részfeladatban $G(N, A_1, A_2)$ S-gráfját. Az S-gráfon a leghosszabb út kereső algoritmussal meghatározhatjuk a részfeladathoz tartozó alsó korlátot, a gráf ütemezési éleit követve meghatározhatjuk a berendezések ütemezéseit és az ütemezésre váró taszkok halmazát. Kevesebb adminisztrációval jár az algoritmus megvalósításában, ha a részfeladat alsó korlát értékét, a berendezések ütemezését és az ütemezésre váró taszkok halmazát a részfeladatban a G(PP) S-gráf mellett redundánsan is tároljuk. Jelölje bound(PP) a PP részfeladat alsó korlátját. Egy berendezés ütemezése megadható a hozzá rendelt taszkok sorrendjével. Az algoritmus során ebbe a taszk sorrendbe új taszkokat szúrunk be. A láncolt lista adatszerkezettel hatékonyan ábrázolható az algoritmusban egy berendezés ütemezése. A PP

részfeladatban az EQLIST(PP) halmaz tartalmazza a berendezések ütemezéseit. Legyen EQLIST$_e$(PP)=L, ahol az (e, L) ∈EQLIST(PP), $e \in E$ és az L pedig az e berendezés ütemezését tartalmazó láncolt lista. Az SOUN(PP) halmaz tartalmazza a részfeladatban még nem ütemezett taszkokat.

A TA-SG algoritmus a *TA-SG* eljárás meghívásával indul. Az eljárás pszeudó kódja a 4.1 ábrán található. Az eljárás feladata az adatstruktúrák inicializálása, az első (gyökér) részfeladat generálása és a nyitott részfeladatok menedzselése a szétválasztási eljárás hívásával. A *TA-SG* eljárás bemenete a $G(N, A_1, \emptyset)$ recept-gráf. A recept-gráf alapján meghatározhatóak az e berendezéshez hozzárendelhető taszkok halmaza, mely halmazt az N_e jelöli ($e \in E$). A gyökér részfeladatnál az EQLIST(PP) üres halmaz, hiszen még egyik berendezéshez se rendeltünk taszkokat, az SOUN(PP) halmaz a recept-gráf taszk-csúcsait tartalmazza, a részfeladat bound(PP) értékét pedig a leghosszabb út kereső algoritmus határozza meg.

A nyitott részfeladatokat a SET halmazban tároljuk, mely halmaz a kezdetben üres. A *current_best* változó tárolja az algoritmus futása során a pillanatnyilag legjobb megoldás végrehajtási idejét, az algoritmus befejezésekor pedig az optimális végrehajtási idő értéket. Mivel az algoritmus indulásakor még nem ismerjük a feladatnak a megoldását, ezért a változó értékét végtelenre állítjuk. A *solution* változóban tároljuk az aktuális legjobb ütemezést.

4.2.2. Szétválasztás eljárás

A TA-SG ütemezési algoritmus szétválasztás eljárása a 4.2 ábrán található. A *TA-szétválasztás* eljárás a paraméterként kapott PP részfeladatból újabb gyermek részfeladatokat generál figyelembe véve az összes aktuális döntési lehetőséget.

A szétválasztási eljárás a döntési lépésében két műveletet hajt végre: kiválaszt egy olyan taszkot, mely még nincsen beütemezve az SOUN(PP) halmazból, majd a kiválasztott taszkot beilleszti egy megfelelő berendezés aktuális ütemezésébe. A kiválasztott taszkot az n változóban tárolja. A taszkot végrehajtható összes lehetséges berendezést hozzárendeljük, azaz befűzzük a berendezés aktuális ütemezésébe az új n taszkot. Az S-gráf n taszkjához tartozó Sn halmaz tartalmazza a taszkhoz rendelhető

procedure *TA-SG*
jelölések:
N_e: az e berendezéssel végrehajtható taszkok halmaza ($e \in E$)
$PP = (G(N, A_1, A_2), bound, EQLIST, SOUN)$: részfeladat
G(PP): a részfeladathoz tartozó S-gráf
bound(PP): a részfeladathoz tartozó *bound* érték
EQLIST(PP)=$\{(e, L) : e \in E$ és L az ütemezést leíró láncolt lista $\}$
SOUN(PP): a részfeladathoz tartozó $SOUN$ halmaz
SET: megoldandó részfeladatokat tartalmazó halmaz
bemenet:
$G(N, A_1, \emptyset)$ recept-gráf
begin
$\forall e \in E, N_e$ halmazok meghatározása;
$SET := \emptyset$;
$current\_best := \infty$;
G(PP):= $G(N, A_1, \emptyset)$;
SOUN(PP):= $\bigcup_{i \in E} N_i$;
EQLIST(PP):= $\emptyset$;
bound(PP):=*TA-korlátozás*(PP);
if bound(PP)$<\infty$
begin
$SET := SET \cup \{PP\}$;
while $SET \neq \emptyset$ **do**
begin
vegyünk ki egy elemet SET-ből, jelöljük PP-vel;
TA-szétválasztás(PP);
end
end
end.
kimenet:
current_best tartalmazza az optimum értékét
solution tartalmazza az optimális ütemezési-gráfot

4.1. ábra. A *TA-SG* eljárás.

berendezéseket. Ha például az ütemezésre kiválasztott taszkot két különböző berendezéssel lehet végrehajtani (az Sn halmaz két elemű), mely berendezések jelenlegi ütemezése már k_1, illetve k_2 darab sorba fűzött taszkot tartalmaz (az EQLIST(PP) halmaz megfelelő elemei k_1 és k_2 elemű láncolt listák), akkor az új taszk valamely berendezéshez való hozzárendelése és befűzése a berendezések taszk végrehajtási sorába $(k_1+1)+(k_2+1)$ darab új gyermek részfeladatot eredményez, mivel egymástól függetlenül mindkét berendezéssel végre lehet hajtani a taszkot. Mindkét berendezés esetén az új ütemezendő taszkot be lehet fűzni valamelyik jelenleg is szereplő taszk elé, vagy pedig a taszk végrehajtási sor legvégére, azaz összesen láncolt lista elemszáma plusz egy új lehetőség áll fent. A gyermek részfeladatokat a $CHILD$ változóval generálja az eljárás. A *korlátozás* eljárás a $CHILD$ részfeladat alsó korlátját határozza meg az ütemezés végrehajtási idejére vonatkozóan. Ha a $CHILD$ részfeladathoz meghatározott alsó korlát kisebb, mint az eddigi legjobb megoldás alsó értéke, akkor a részfeladat bekerül a SET halmazba.

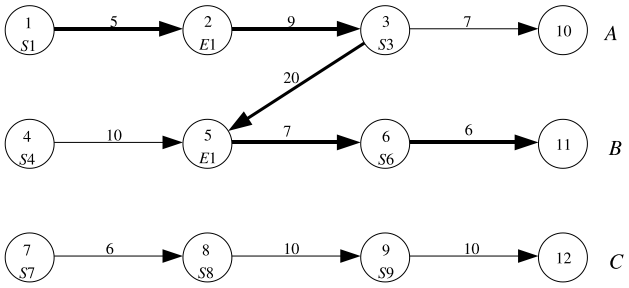
Az EQ-SG algoritmusban a szülő részfeladat S-gráfjának leghosszabb útja része a gyermek részfeladat S-gráfjának, hiszen a gyermek részfeladat S-gráfjában ugyanazok az élek szerepelnek, legfeljebb recept-élek értékei növekedhetnek. Ezért teljesül, hogy a gyermek részfeladat alsó korlátja nem kisebb, mint a szülő részfeladaté. A TA-SG algoritmus során a gyermek részfeladatban a berendezések ütemezésének változásakor ütemezési-él törlődik a szülő részfeladat S-gráfjához képest, ami miatt előfordulhat, hogy a szülő részfeladat S-gráfjának leghosszabb útja nem szerepel a gyermek részfeladat S-gráfjában.

Tegyük fel, hogy a PP részfeladatban az $E1 \in E$ berendezésnek ütemezését leíró láncolt lista alapján a berendezés először a 2-es, majd az 5-ös taszkot hajtja végre, azaz az EQLIST$(PP)_{E1}$=2 $\longrightarrow$ 5. A két taszk között 20 időegységnyi váltási időre van szüksége az $E1$ berendezésnek. A PP részfeladat ütemezése a 4.3 ábrán látható. Az S-gráf leghosszabb útja 47.

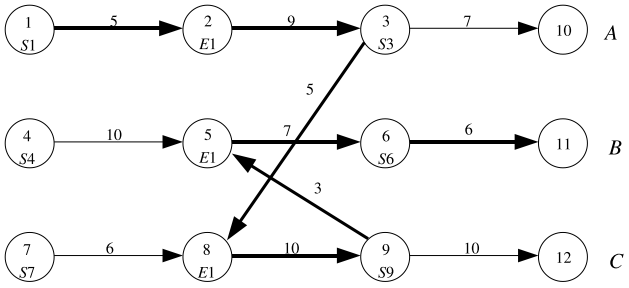
Tegyük fel, hogy az $S8 = \{E1\}$ és a PP részfeladatból a *TA-szétválasztás* eljárás a 8-as taszkot választja ki következőnek ütemezendőnek ($n = 8$). Ekkor az eljárás során három gyermek részfeladatot kaphatunk, amiket a $CHILD$ változó segítségével generálunk. A gyermek részfeladatok $E1$ berendezéshez tartozó ütemezései a következők

```
procedure TA-szétválasztás(PP)
jelölések:
    EQLIST(PP)_e=L, ahol (e,L) ∈EQLIST(PP), e ∈ E
begin
    if BOUND(PP)<current_best
      begin
        legyen n ∈ SOUN(PP);
        SOUN(PP):=SOUN(PP)\{n};
        for all e ∈ Sn
          begin
            for i := 1 to |EQLIST(PP)_e| + 1
              begin
                CHILD := PP;
                az n taszk beszúrása az EQLIST(CHILD)_e i-edik pozíciójába;
                G(CHILD) S-gráf módosítása az EQLIST(CHILD)_e alapján;
                bound(CHILD):=TA-korlátozás(CHILD);
                if bound(CHILD)<current_best
                  begin
                    if SOUN(CHILD) ≠ ∅
                        SET := SET ∪ {CHILD};
                    else
                      current_best és solution módosítása;
                  end
              end
          end
      end
end.
```

4.2. ábra. A taszk alapú döntési stratégiát megvalósító *TA-szétválasztás* eljárás a TA-SG algoritmus számára.



4.3. ábra. A szülő részfeladat ütemezése (leghosszabb út: 47).



4.4. ábra. A gyermek részfeladat ütemezése (leghosszabb út: 45).

lehetnek:

1. EQLIST$(CHILD)_{E1}$=8 $\longrightarrow$ 2 $\longrightarrow$ 5
2. EQLIST$(CHILD)_{E1}$=2 $\longrightarrow$ 8 $\longrightarrow$ 5
3. EQLIST$(CHILD)_{E1}$=2 $\longrightarrow$ 5 $\longrightarrow$ 8.

Az 1. és 3. gyermek részfeladathoz tartozó S-gráf tartalmazza a szülő részfeladat S-gráfjának leghosszabb útját, így ezeknek a részfeladatoknak az alsó korlátja nem lehet kisebb, mint a szülő részfeladaté. A 2. gyermek részfeladat S-gráfja azonban nem tartalmazza a szülő részfeladat leghosszabb útját az ütemezési-élek megváltozása miatt (lásd a 4.4. ábrát). Ahogy a példa is mutatja, az ütemezési-élek változásával a gyermek részfeladat S-gráfjának leghosszabb útja rövidebb is lehet, mint a szülő részfeladaté.

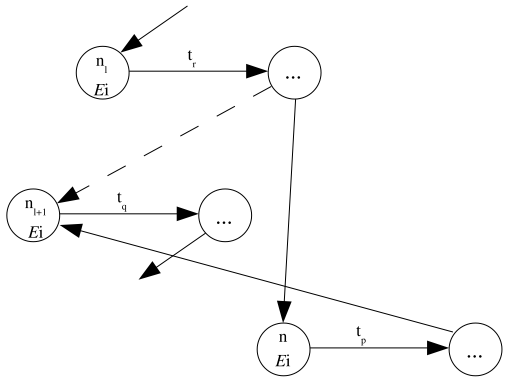
Az előző példa azt jelzi, hogy a TA-SG algoritmus nem alkalmazható bármelyik ütemezési feladat megoldására. Ha az ütemezési feladat megoldása során a TA-SG algoritmus olyan gyermek részfeladatokhoz jut, melyeknek az alsó korlátja kisebb, mint a szülő részfeladaté, akkor a TA-SG algoritmus futása során nem biztosított az optimális megoldás megtalálása a részfeladatok esetleges kizárása miatt. A következő állítások a TA-SG algoritmussal megoldható feladatok osztályát ismertetik.

4.2.1. Tétel. *Ha az ütemezési feladat receptjének váltási idejeire teljesül a háromszög egyenlőtlenség, azaz bármely berendezés esetén teljesül, hogy* $t_{ij} \leq t_{ik} + t_{kj}$*, ahol* t_{ij}, t_{ik}, t_{kj} *a berendezés váltási ideje az* i*,*j*, az* i*,*k *és a* k*,*j *taszkok között, akkor a szülő részfeladatból a TA-szétválasztás eljárással előállított bármely gyermek részfeladatnak az alsó korlátjai nem kisebbek, mint a szülő részfeladat alsó korlátja.*

4.2.1. Bizonyítás. *Tegyük fel, hogy a PP részfeladat S-gráfját a G(PP), alsó korlátját a bound(PP) jelöli. A CHILD gyermek részfeladat létrehozásakor a következő esetek állhatnak fenn:*

(i) A CHILD részfeladatban az új, ütemezésre kiválasztott taszk ütemezéséhez nem kell a PP részfeladat ütemezési-éleit törölni. Ez az eset akkor áll fenn, ha az ütemezésre kiválasztott taszkot egy berendezés első, vagy utolsónak végrehajtandó taszkjaként ütemezzük. Ebben az esetben a berendezés végrehajtási sorába befűzhető a taszk a G(PP) S-gráf ütemezési-éleinek módosítása nélkül. Mivel a gyermek részfeladatok S-gráfjai tartalmazzák a szülő részfeladat S-gráfjának leghosszabb útját, ezért a gyermek részfeladatok alsó korlátjai nem kisebbek, mint a szülő részfeladaté.

(ii) A CHILD részfeladatban az új, ütemezésre kiválasztott taszk ütemezéséhez törölni kell ütemezési-éleket a PP részfeladat ütemezési-éleiből. Tegyük fel, hogy a kiválasztott berendezés ütemezése a PP részfeladatban az $n_1 \longrightarrow n_2 \longrightarrow ... \longrightarrow n_k$ *taszkok láncolt listája tartalmazza. Az ütemezendő* n *taszkot az* l*. taszk után ütemezzük be, mely ütemezés az* $n_1 \longrightarrow n_2 \longrightarrow ... \longrightarrow n_l \longrightarrow n \longrightarrow n_{l+1} \longrightarrow ... \longrightarrow n_k$ *láncolt listával írható le.*



4.5. ábra. A *TA-szétválasztás* eljárás során új taszk ütemezése a gyermek részfeladatban.

Az n taszk ütemezése az n_l és n_{l+1} taszkok közötti ütemezési-él törlésével és az n_l és n és az n n_{l+1} taszkok közötti ütemezési-élek bevezetésével jár. Ha a törölt ütemezési-él nem a része a PP részfeladat leghosszabb útjának, akkor a PP részfeladat leghosszabb útja része a gyermek részfeladat S-gráfjának, ezért a gyermek részfeladat alsó korlátja nem lehet kisebb, mint a szülőé volt. Ha a törölt él a leghosszabb út része volt, akkor NIS esetén a 4.5 ábra szemlélteti az ütemezési-élek változását. A szaggatott vonallal jelölt él helyett új ütemezési éleken lehet az n_l és n_{l+1} csúcsok között haladni. Az új út hossza az ütemezési élekre fennálló egyenlőtlenség tulajdonság miatt legalább olyan hosszú, mint a szaggatott ütemezési-élen vezető útnak, ezért a gyermek részfeladat alsó korlátja nem csökkenhet a szülő részfeladat korlátjához képest.

4.2.2. Tétel. *Ha egy részfeladat S-gráfja kört tartalmaz, akkor a TA-SG algoritmus „taszk alapú szétválasztási döntései" során a részfeladatból nem kaphatunk körmentes gyermek részfeladatot.*

4.2.2. Bizonyítás. *Az állítás azt mondja ki, hogy nem megvalósítható részütemezésből „taszk alapú szétválasztási döntések" alapján csak nem megvalósítható részütemezéseket kapunk. Kört tartalmazó S-gráfból csak él kivételével kaphatunk körmentes S-gráfot. A 4.2.1 tétel bizonyítása alapján a 4.5 ábra esetén törlünk ütemezési-élt az*

S-gráfból. Az ábra alapján a szaggatott él törlésével nem szűnik meg az él végpontjai közötti út, így a kört tartalmazó S-gráfból „taszk alapú döntéssel" csak kört tartalmazó gyermek részfeladatot kaphatunk.

Ha az ütemezési feladat váltási idejeire teljesül a háromszög egyenlőtlenség, akkor a 4.2.1 és 4.2.2 tételek alapján a TA-SG algoritmus a szétválasztás és korlátozás módszerének megfelelően helyesen járja be a keresési teret. Csak azokat a részfeladatokat zárja ki a vizsgálatból, melyek csak nem megvalósítható ütemezéshez vezetnek, vagy bizonyítottan nem optimális ütemezéseket kaphatunk belőlük. A 4.2.1 és 4.2.2 tételek feltételei ipari ütemezési feladatok esetén teljesülnek, gyakorlati feladatok megoldása során nem jelent megkötést.

4.2.3. Korlátozás eljárás

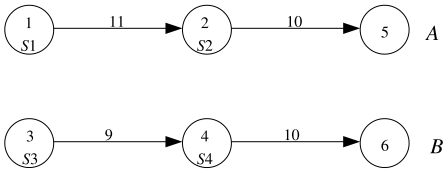
A taszk alapú döntéseket alkalmazó szétválasztási eljárás nem befolyásolja az EQ-SG algoritmus korlátozás eljárását, ezért változtatás nélkül használhatóak az EQ-SG algoritmusban bevezetett körkereső és leghosszabb út kereső algoritmust használó *EQ-korlátozás* eljárás (3.7 ábra). A korlátozás eljárásban a részfeladat megvalósíthatóságát az S-gráf körmentességének vizsgálatával döntjük el. Amennyiben a részfeladat megvalósítható ütemezést jelöl, akkor a leghosszabb út kereső algoritmussal számolhatjuk ki a részfeladatból elérhető ütemezések végrehajtási idejének alsó korlátját.

4.2.4. TA-SG algoritmus működésének bemutatása

A TA-SG algoritmus működését ütemezési feladatok megoldásán keresztül szemléltetem. Mindegyik feladatban az NIS tárolási stratégiát kell teljesíteni.

4.1. feladat

A 4.1. feladattal a célom az EQ-SG és a TA-SG algoritmusokkal bejárt keresési fák összehasonlítása. Az algoritmusok működésének szemléltetésére tekintsük egy olyan ütemezési feladatot, melyben az A és B termékekhez tartozó receptek alapján minden taszkot pontosan egy berendezéssel lehet végrehajtani, azaz nincsen olyan



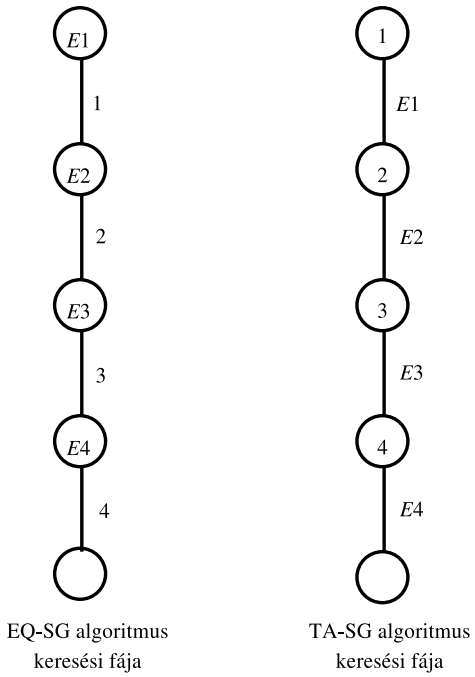
4.6. ábra. 4.1. feladat feladat recept-gráfja.

berendezés, amivel több taszkot el lehet végezni. A 4.6 ábrán látható recept-gráf egy ilyen ütemezési feladat receptjét ábrázolja. Legyen az $S1 = \{E1\}, S2 = \{E2\}, S3 = \{E3\}$ és $S4 = \{E4\}$ a taszkokhoz hozzárendelhető berendezések halmazai. A taszkok végrehajtásához szükséges időmennyiségek fel vannak tüntetve a recept-gráf élein.

Ez a lehető legegyszerűbb feladat, mert a szétválasztás eljárás során nem keletkezhet olyan részfeladat ahol ütemezési döntési alternatívák állnak a rendelkezésre. A feladat megoldása egyértelmű. A keresési fák egyedül az algoritmusok lefutásában, a végrehajtott döntések sorrendjében változhat. Az EQ-SG algoritmus futása a berendezés alapú ütemezés esetén a berendezés kiválasztás sorrendjétől, a TA-SG algoritmus esetén pedig az ütemezendő taszkok kiválasztásának sorrendjétől függ.

A 4.7 ábra az EQ-SG algoritmus berendezés alapú döntési stratégiájával és a TA-SG algoritmus taszk alapú döntési stratégiájával bejárt keresési fáikat mutatja. A berendezés alapú döntés során tegyük fel, hogy a berendezések kiválasztása $E1$, $E2$, $E3$, $E4$ sorrendben történik. A berendezések kiválasztásának sorrendje miatt a taszkokat 1, 2, 3, 4 sorrendben ütemezzük a megfelelő berendezéshez. A feladat jellege miatt minden berendezéshez pontosan egy taszkot lehet beütemezni, ezért minden szülő részfeladatból egy új gyermek részfeladat keletkezik. A kiválasztott berendezés a fa csúcspontjaiban szerepel, a berendezéshez beütemezett taszk azonosítója a fa élének címkéjén látható.

A TA-SG algoritmus taszk alapú döntési stratégiája során a nem ütemezett taszkok kiválasztásának sorrendjében épül fel a keresési fa. A kiválasztott taszkhoz rendelünk hozzá egy megfelelő berendezést. A TA-SG algoritmus a taszkokat 1, 2, 3, 4 sorrendben választja ki ütemezésre és rendeli hozzájuk az $E1$, $E2$, $E3$, $E4$ berendezéseket. Az ábrán a kiválasztott taszkok a gráf csúcsaiban, a taszkokhoz rendelt és



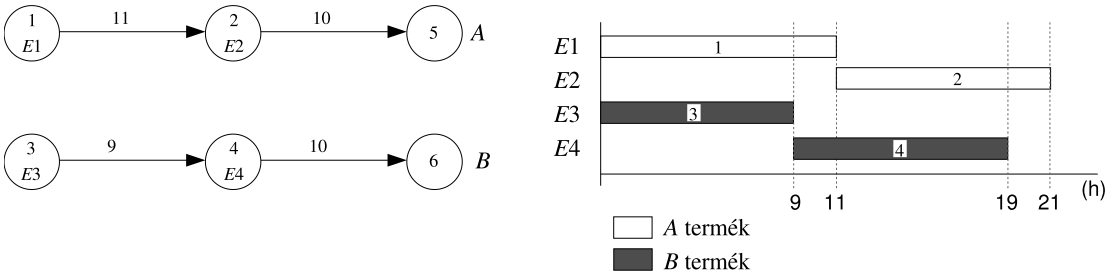
4.7. ábra. A 4.1. feladat EQ-SG és TA-SG algoritmusokkal bejárt keresési fái.

ütemezett berendezések a gráf élein szerepelnek. Mivel nem lehet olyan berendezés, mely több taszkot is végrehajt, ezért a berendezés taszk végrehajtási sorát nem kell ütemezni ebben a feladatban.

A bemutatott algoritmusok mindegyike a használt döntési stratégiától függetlenül ugyanolyan jellegű keresési fát járnak be. Az ütemezésre kiválasztott berendezések illetve taszkok sorrendjétől függetlenül ugyanazt az optimális ütemezést adja megoldásnak az EQ-SG és a TA-SG algoritmus is. A 4.8 ábra a feladat optimális megoldásának ütemezési-gráfját és Gantt-diagrammját ábrázolja. Ennél a feladatnál a két algoritmus futási ideje körülbelül ugyanannyi (az algoritmusok implementációjától függ).

4.2. feladat

A 4.2. feladatban az A és B termékeket kell előállítani az $E1$, $E2$, $E3$ és $E4$ berendezéseket használva. A 4.1 táblázat tartalmazza a termékek előállítását leíró recepteket.



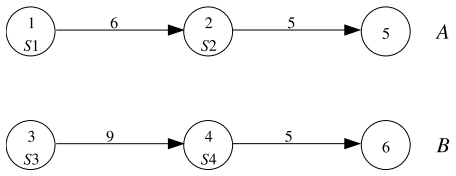
4.8. ábra. A 4.1. feladat optimális ütemezését ábrázoló ütemezési-gráf és Gantt-diagramm.

4.1. táblázat. A 4.2. feladat receptje.

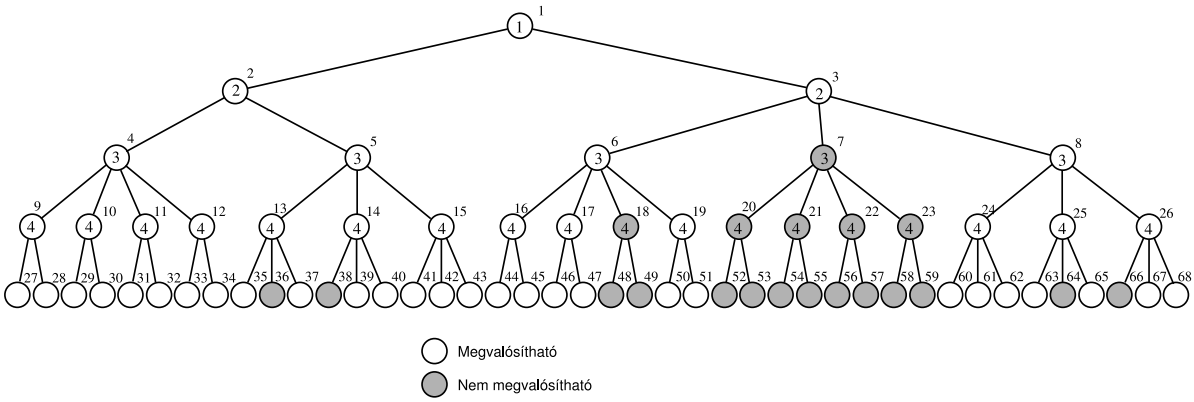
Taszk	A termék		B termék	
	Berendezés	Idő (h)	Berendezés	Idő (h)
1	$E1$	8	$E1$	9
	$E2$	6	$E2$	11
2	$E2$	15	$E3$	5
	$E3$	5	$E4$	7

A 4.9 ábrán található az ütemezési feladat recept-gráfja. A recept alapján a recept-gráfban az $S1 = \{E1, E2\}$, $S2 = \{E2, E3\}$, $S3 = \{E1, E2\}$ és $S4 = \{E3, E4\}$.

A TA-SG algoritmus működésének szemléltetéséhez tegyük fel, hogy a *TA-szét-választás* eljárás során a taszkokat az 1, 2, 3, 4 sorrendben választjuk ki ütemezésre. A 4.10 ábra mutatja az ebben az esetben bejárható teljes keresési fát. A keresési fa mérete miatt a csúcsaihoz tartozó rész ütemezéseket a 4.2 táblázat tartalmazza



4.9. ábra. A 4.2. feladat recept-gráfja.



4.10. ábra. A 4.2. feladat teljes leszámolási keresési fája.

(lásd a fa csúcsainak jobb felső részén elhelyezett azonosítók és a táblázat sorainak számozása). A 4.2 táblázat minden sora egy-egy rész ütemezést jelöl, a rész ütemezés tartalmazza a berendezésekhez rendelt taszkokat és a taszkok sorrendjét és a részfeladathoz számított alsó korlát értékét. Például a 10-es azonosítójú részfeladat alapján a TA-SG algoritmus taszk alapú döntései során az $E1$ berendezés először a 3-as, majd az 1-es taszkot hajtja végre, az $E2$ berendezés a 2-es taszkot hajtja végre, míg az $E3$ és $E4$ berendezésekhez nincsenek taszkok rendelve. A részfeladatban a 4-es taszkhoz még nincsen berendezés hozzárendelve.

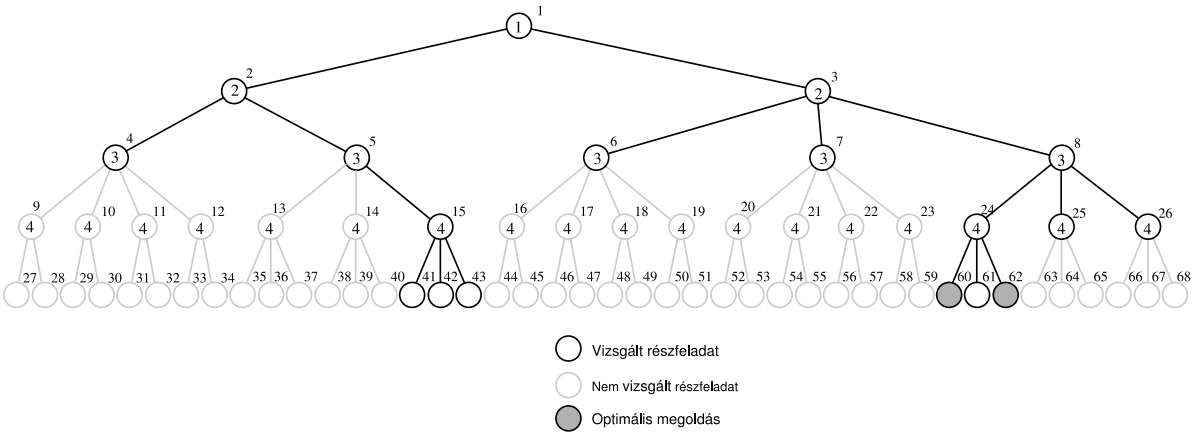
A teljes keresési fa a feladat méreteit és az összes lehetséges ütemezést szemlélteti, a szétválasztás és korlátozás elvén működő ütemező algoritmusok azonban nem járják be az egész keresési fát. Vannak olyan részfeladatok, melyeket nem szükséges megvizsgálni. Ha egy részfeladat nem megvalósítható ütemezést jelöl, akkor nem kell tovább vizsgálni, hiszen a belőle kapott összes gyermek részfeladat és ütemezési-gráf is nem megvalósítható marad. Ha valamely részfeladat alsó korlátja nem kisebb, mint az addig megtalált legjobb ütemezés végrehajtási ideje, akkor ezt a részfeladatot se kell tovább vizsgálni, hiszen a részfeladatból nem kaphatunk jobb végrehajtási idejű ütemezést, mint az eddigi legjobb megoldásunk. A 4.11 ábra a TA-SG algoritmus futása során ténylegesen bejárt keresési fát szemlélteti. A bejárt keresési fa a mélységi keresést megvalósító algoritmus során keletkezett.

A kapott optimális ütemezés végrehajtási ideje 16 óra. Az optimális ütemezés a

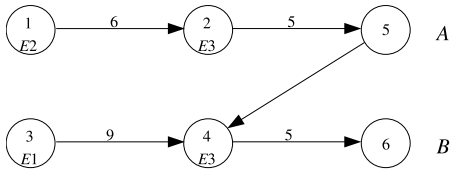
4.2. táblázat. A 4.10 ábra keresési fájának csúcsaihoz tartozó ütemezések.

Csúcs	Alsó korlát	Berendezés				Csúcs	Alsó korlát	Berendezés			
		$E1$	$E2$	$E3$	$E4$			$E1$	$E2$	$E3$	$E4$
1	14	-	-	-	-	35	22	1,3	-	2,4	-
2	14	1	-	-	-	36	∞^*	1,3	-	4,2	-
3	14	-	1	-	-	37	24	1,3	-	2	4
4	23	1	2	-	-	38	∞^*	3,1	-	2,4	-
5	14	1	-	2	-	39	22	3,1	-	4,2	-
6	21	-	1,2	-	-	40	22	3,1	-	2	4
7	∞^*	-	2,1	-	-	41	18	1	3	2,4	-
8	14	-	1	2	-	42	21	1	3	4,2	-
9	23	1,3	2	-	-	43	18	1	3	2	4
10	32	3,1	2	-	-	44	21	3	1,2	4	-
11	39	1	2,3	-	-	45	21	3	1,2	-	4
12	26	1	3,2	-	-	46	37	-	1,2,3	4	-
13	22	1,3	-	2	-	47	39	-	1,2,3	-	4
14	22	3,1	-	2	-	48	∞^*	-	1,3,2	4	-
15	16	1	3	2	-	49	∞^*	-	1,3,2	-	4
16	21	3	1,2	-	-	50	32	-	3,1,2	4	-
17	37	-	1,2,3	-	-	51	32	-	3,1,2	-	4
18	∞^*	-	1,3,2	-	-	52	∞^*	3	2,1	4	-
19	32	-	3,1,2	-	-	53	∞^*	3	2,1	-	4
20	∞^*	3	2,1	-	-	54	∞^*	-	2,1,3	4	-
21	∞^*	-	2,1,3	-	-	55	∞^*	-	2,1,3	-	4
22	∞^*	-	2,3,1	-	-	56	∞^*	-	2,3,1	4	-
23	∞^*	-	3,2,1	-	-	57	∞^*	-	2,3,1	-	4
24	14	3	1	2	-	58	∞^*	-	3,2,1	4	-
25	22	-	1,3	2	-	59	∞^*	-	3,2,1	-	4
26	22	-	3,1	2	-	60	16	3	1	2,4	-
27	23	1,3	2	4	-	61	19	3	1	4,2	-
28	24	1,3	2	-	4	62	16	3	1	2	4
29	32	3,1	2	4	-	63	22	-	1,3	2,4	-
30	32	3,1	2	-	4	64	∞^*	-	1,3	4,2	-
31	39	1	2,3	4	-	65	24	-	1,3	2	4
32	41	1	2,3	-	4	66	∞^*	-	3,1	2,4	-
33	26	1	3,2	4	-	67	22	-	3,1	4,2	-
34	26	1	3,2	-	4	68	22	-	3,1	2	4

* nem megvalósítható



4.11. ábra. A TA-SG algoritmus során bejárt keresési fa.

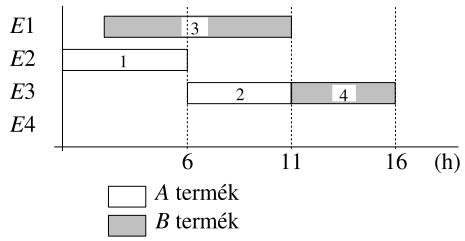


4.12. ábra. A 4.2. feladat egy optimális megoldásának ütemezési-gráfja.

keresési fában a 60-as azonosítójú részfeladathoz tartozik, mely ütemezésben az $E1$ berendezés a 3-as, az $E2$ berendezés az 1-es, az $E3$ berendezés pedig először a 2-es, majd a 4-es taszkot hajtja végre. A 62-es azonosítójú részfeladatot is ellenőrzi az algoritmus, mely részfeladat szintén megoldása a feladatnak és emellett ez az ütemezés is optimális, azonban az algoritmus csak az elsőnek megtalált optimális megoldást tárolja. A 60-as azonosítójú optimális ütemezéshez tartozó ütemezési-gráf és Gantt diagramm a 4.12 és a 4.13 ábrákon találhatóak.

4.3. feladat

A 4.3. feladatban – egy új termékkel tovább növelve a 4.2. feladat méretét – az A,B és C termékeket kell előállítani az $E1$, $E2$, $E3$ és $E4$ berendezéseket használva. A 4.3. táblázat tartalmazza a termékek előállítását leíró recepteket. Az A termékből 3 batch



4.13. ábra. A 4.2. feladat egy optimális megoldásának Gantt-diagrammja.

4.3. táblázat. A 4.3. feladat receptje.

Taszk	A termék		B termék		C termék	
	Berendezés	Idő (h)	Berendezés	Idő (h)	Berendezés	Idő (h)
1	$E1$	8	$E1$	9	$E1$	7
	$E2$	6	$E2$	11	$E2$	7
2	$E2$	15	$E3$	5	$E3$	4
	$E3$	5	$E4$	7		

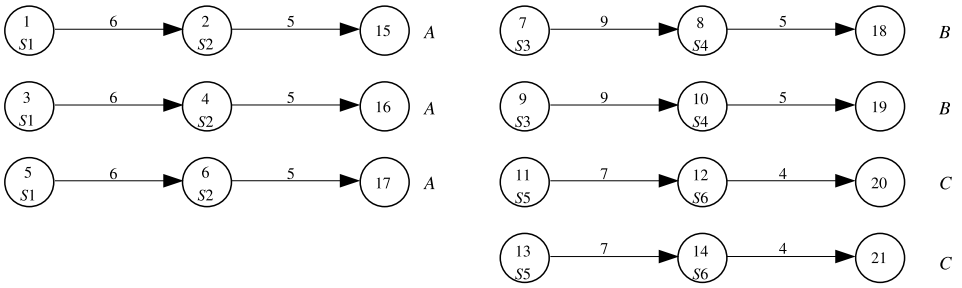
mennyiséget, a B és C termékekből pedig 2-2 batch mennyiséget kell beütemezni. A 4.14. ábra mutatja a feladat recept-gráfját. A recept alapján a recept-gráfban az $S1 = \{E1, E2\}$, $S2 = \{E2, E3\}$, $S3 = \{E1, E2\}$, $S4 = \{E3, E4\}$, $S5 = \{E1, E2\}$ és $S6 = \{E3\}$.

A feladat optimális ütemezésének végrehajtási ideje 33 óra. Egy optimális ütemezést leíró ütemezési-gráf és Gantt-diagramm a 4.15. és a 4.16. ábrákon találhatóak. A TA-SG algoritmus 0,21 másodperc alatt találta meg a feladat optimális megoldását[1].

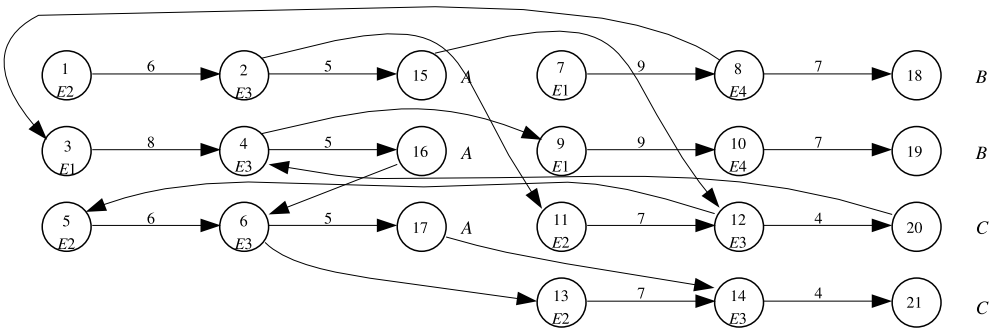
4.3. Több batch egyidejű kezelése

A Holczinger és társai által közölt segéd-él technikát bemutattam az EQ-SG algoritmusnál [34]. A recept-gráf segéd-élekkel való kiegészítése független az ütemező algoritmusban használt szétválasztási eljárás típusától, ezért természetesen független

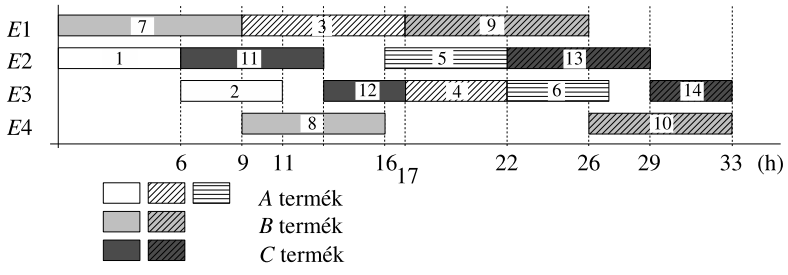
[1] Az algoritmus egy 1,5 GByte DDR RAM-mal felszerelt, Intel Pentium 4 3.06GHz órajelű PC-n futott.



4.14. ábra. A 4.3. feladat recept-gráfja.



4.15. ábra. A 4.3. feladat egy optimális ütemezési-gráfja.



4.16. ábra. A 4.3. feladat egy optimális ütemezésének Gantt-diagrammja.

4.4. táblázat. A 4.4. feladat receptje.

Taszk	A termék		B termék		C termék		D termék	
	Beren-dezés	Idő (h)	Beren-dezés	Idő (h)	Beren-dezés	Idő (h)	Beren-dezés	Idő (h)
1	$E1$	10	$E1$	9	$E1$	7	$E2$	9
	$E2$	6	$E3$	14	$E2$	7		
2	$E2$	15	$E3$	10	$E3$	4	$E3$	4
	$E3$	5	$E4$	7			$E4$	8
3					$E3$	6	$E3$	7
					$E5$	10	$E5$	10

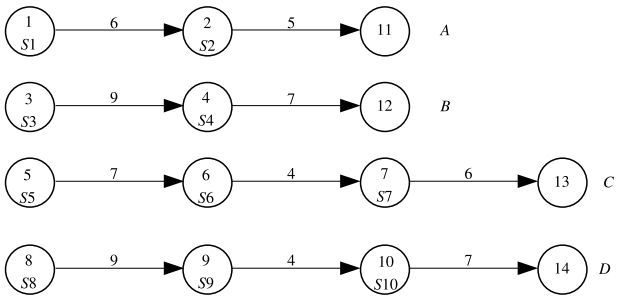
a használt döntési stratégiától is. Az alapalgoritmusnál megfogalmazott segéd-élekre vonatkozó állítások a *TA-szétválasztás* eljárást használó ütemező algoritmus esetén is érvényesek.

Adott termékekből több batchet tartalmazó ütemezési feladatok megoldásával szemléltetem a segéd-él technika hatékonyságát összehasonlítva a segéd-éleket nem tartalmazó feladat megoldásának sebességével. A megoldások megtalálásához a TA-SG algoritmust használom.

4.4. feladat

A 4.4. feladatban az A, B, C és D termékek receptjei alapján felépített különböző méretű ütemezési feladatokat kell megoldanunk. A feladatokban különböző mennyiségű termékeket kell ütemezni az eredeti és a segéd-éleket tartalmazó recept-gráfokból kiindulva. A gyártást ismertető receptet a 4.4 táblázat tartalmazza. Az 1-1 batchnyi A, B, C és D terméket ábrázoló recept-gráf a 4.17 ábrán látható. A recept-gráfban az $S1 = \{E1, E2\}$, $S2 = \{E2, E3\}$, $S3 = \{E1, E3\}$, $S4 = \{E3, E4\}$, $S5 = \{E1, E2\}$, $S6 = \{E3\}$, $S7 = \{E3, E5\}$, $S8 = \{E2\}$, $S9 = \{E3, E4\}$ és $S10 = \{E3, E5\}$.

A különböző méretű feladatokra kapott futási eredményeket a 4.5 táblázat tartalmazza. A táblázat tartalmazza a segéd-éleket használó és segéd-éleket nem használó TA-SG algoritmus futási eredményeit. Az 1-es sorszámú feladatban, mivel minden



4.17. ábra. A 4.4. feladat recept-gráfja.

termékből 1 batchnyit kell előállítani, ezért nem lehet a segéd-élekkel gyorsítani a feladat megoldását. Kis méretű feladatoknál a futási időben nincsen szignifikáns eltérés. A feladatméret növekedésével a futási eredmények azt mutatják, hogy segéd-élek alkalmazásával a recept-gráfban gyorsabban kapunk bizonyítottan optimális megoldást. A 11-es feladatnál a segéd-él nélküli receptre az algoritmus már nem találta meg az optimális megoldást 1 óra alatt[2].

4.4. Az EQ-SG és a TA-SG algoritmusok viselkedésének az összehasonlítása

Az EQ-SG és a TA-SG algoritmusok ugyanannak a feladatosztálynak adják meg egy optimális megoldását. Az ütemezőkkel kapott megoldások optimálisak (ugyanolyan végrehajtási idejűek), a kapott ütemezések azonban különbözhetnek (általában több optimális megoldása van egy ütemezési feladatnak). Ebben a részben bemutatok három ütemezési feladatot, a 4.5 feladatot körülbelül azonos hatékonysággal oldják meg az ütemező algoritmusok, a 4.6 feladatot a TA-SG, a 4.7 feladatot az EQ-SG algoritmus oldja meg hatékonyabban. Az implementált algoritmusokkal lehetőség van az összes optimális ütemezési-gráf meghatározására. Mindkét algoritmus ugyanazokat az optimális ütemezési-gráfokat generálta a vizsgálatok során.

[2]Az algoritmus egy 1,5 GByte DDR RAM-mal felszerelt, Intel Pentium 4 3.06GHz órajelű PC-n futott.

4.5. táblázat. A 4.4. feladat megoldása során a TA-SG algoritmussal kapott futási eredmények.

Sorszám	Batch-ek száma				Optimum	Segéd-élek nélkül	Segéd-élekkel
	A	B	C	D	(h)	(sec)	(sec)
1	1	1	1	1	27	0,05	0,05
2	1	1	1	2	34	0,13	0,06
3	1	1	2	1	33	0,14	0,12
4	1	1	2	2	39	0,97	0,26
5	1	2	1	1	31	0,07	0,07
6	1	2	1	2	36	0,59	0,20
7	2	1	1	1	31	0,09	0,06
8	2	2	2	2	43	9,55	2,21
9	3	2	2	2	47	62,86	6,31
10	3	3	2	2	49	301,64	12,99
11	3	3	3	2	55	–	90,58

4.5. feladat

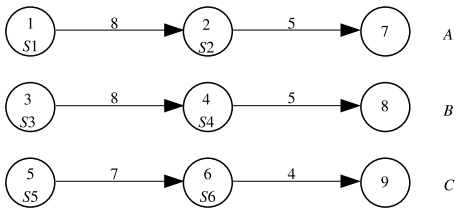
A 4.5. feladat megoldásával taszk alapú és a berendezés alapú döntéseket használó ütemezési algoritmusok futási eredményeit hasonlítom össze különböző méretű ütemezési feladatok megoldásával. A 4.6 táblázat tartalmazza az ütemezési feladatokban előállítandó termékek recept leírását. Az A, B és C termékeket 1-1 batchben előállító recept-gráf a 4.18. ábrán található. A recept-gráfban az $S1 = \{E1, E2\}$, $S2 = \{E2, E4\}$, $S3 = \{E1, E2\}$, $S4 = \{E3, E4\}$, $S5 = \{E1, E2\}$ és $S6 = \{E3\}$.

A 4.7 táblázat tartalmazza a recept-gráf alapján felépített különböző méretű ütemezési feladatok futási eredményeit az EQ-SG és a TA-SG algoritmusokkal megoldva. Az algoritmusok futási idejében nincsen jelentős különbség, közel azonos hatékonysággal oldják meg a kitűzött ütemezési feladatokat[3].

[3] Az algoritmus egy 1,5 GByte DDR RAM-mal felszerelt, Intel Pentium 4 3.06GHz órajelű PC-n futott.

4.6. táblázat. A 4.5. feladat receptje.

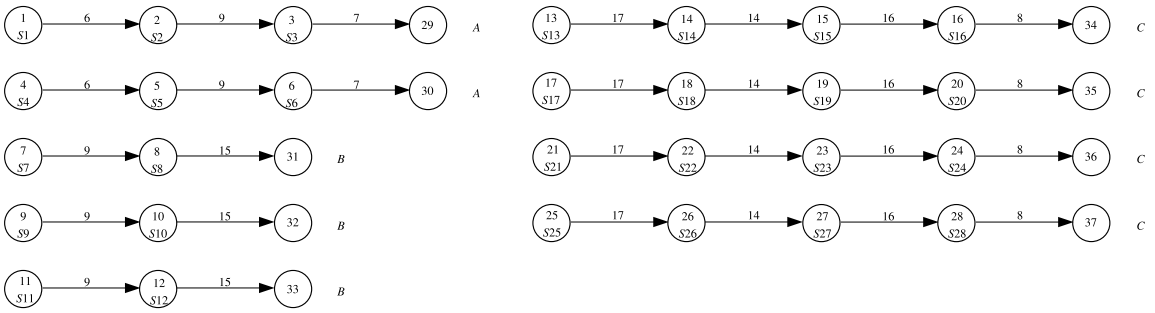
Taszk	A termék		B termék		C termék	
	Berendezés	Idő (h)	Berendezés	Idő (h)	Berendezés	Idő (h)
1	$E1$	8	$E1$	8	$E1$	8
	$E2$	6	$E2$	11	$E2$	7
2	$E2$	15	$E3$	5	$E3$	4
	$E4$	5	$E4$	7		



4.18. ábra. A 4.5. feladat recept-gráfja (1-1 batchnyi minden termékből).

4.7. táblázat. A 4.5. feladat megoldása során kapott futási eredmények.

Sorszám	Batch-ek száma			Optimum	EQ-SG	TA-SG
	A	B	C	(h)	(sec)	(sec)
1	1	1	1	17	0,01	0,01
2	2	2	1	24	0,10	0,24
3	3	2	2	30	0,51	0,34
4	3	3	2	36	1,06	0,75
5	3	3	3	37	2,37	1,99
6	4	3	3	42	13,32	12,77
7	4	4	3	44	30,92	24,00



4.19. ábra. A 4.6. feladat recept-gráfja.

4.6. feladat

Nagy méretű feladatok esetén a keresési tér bejárási módjától függően az ütemező algoritmusok futási idejei között nagy különbségek lehetnek. Az elvégzett tesztek alapján a TA-SG algoritmus az EQ-SG-hez képest a „kiegyenlített" ütemezési feladatok során találja meg hamarabb a feladatok optimális megoldását. A „kiegyenlített" ütemezési feladatok optimális ütemezésében a berendezések közel azonos mennyiségben terheltek.

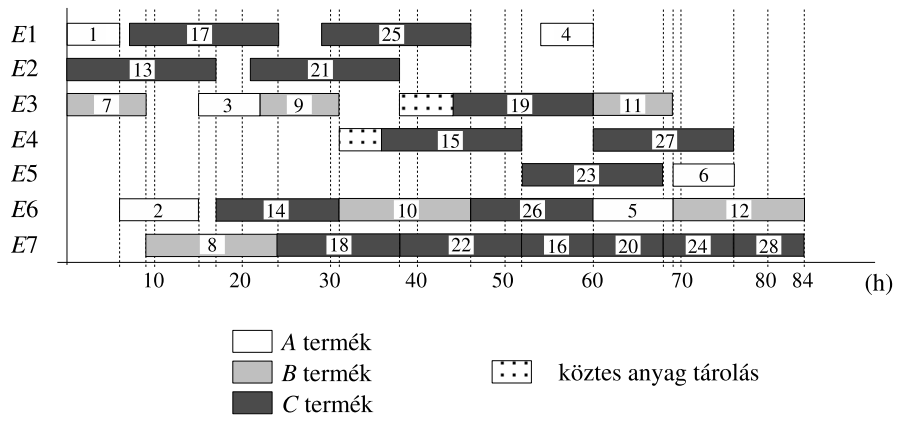
A 4.8 táblázat a feladat receptjét tartalmazza. A feladatban az A termékből 2, a B termékből 3, a C termékből 2 batchnyi mennyiséget kell ütemezni. A 4.19 ábra a feladat recept-gráfját tartalmazza. A recept-gráfban az $S1 = S4 = \{E1, E2\}$, $S2 = S5 = \{E6, E7\}$, $S3 = S6 = \{E3, E4, E5\}$, $S7 = S9 = S11 = \{E3, E4, E5\}$, $S8 = S10 = S12 = \{E6, E7\}$, $S13 = S17 = S21 = S25 = \{E1, E2\}$, $S14 = S18 = S22 = S26 = \{E6, E7\}$, $S15 = S19 = S23 = S28 = \{E3, E4, E5\}$ és $S16 = S20 = S24 = S28 = \{E6, E7\}$.

Az optimális ütemezés végrehajtási ideje 84 óra, mely ütemezés Gantt-diagrammját a 4.20 ábrán található. Az optimális ütemezést vizsgálva láthatjuk, hogy a berendezések többsége azonos mértékben leterhelt. A TA-SG algoritmussal a feladat megoldására 1511 másodperc kellett, ugyanazon a PC-n az EQ-SG algoritmussal 12 óra alatt nem kaptuk meg a feladat optimális megoldását[4].

[4]Az algoritmus egy 1,5 GByte DDR RAM-mal felszerelt, Intel Pentium 4 3.06GHz órajelű PC-n futott.

4.8. táblázat. A 4.6. feladat receptje.

Taszk	A termék		B termék		C termék	
	Berendezés	Idő (h)	Berendezés	Idő (h)	Berendezés	Idő (h)
1	$E1$	6	$E3$	9	$E1$	17
	$E2$	6	$E4$	9	$E2$	17
			$E5$	9		
2	$E6$	9	$E6$	15	$E6$	14
	$E7$	9	$E7$	15	$E7$	14
3	$E3$	7			$E3$	16
	$E4$	7			$E4$	16
	$E5$	7			$E5$	16
4					$E6$	8
					$E7$	8



4.20. ábra. A 4.6. feladat egy optimális ütemezésének Gantt-diagrammja.

4.9. táblázat. A 4.7. feladat receptje.

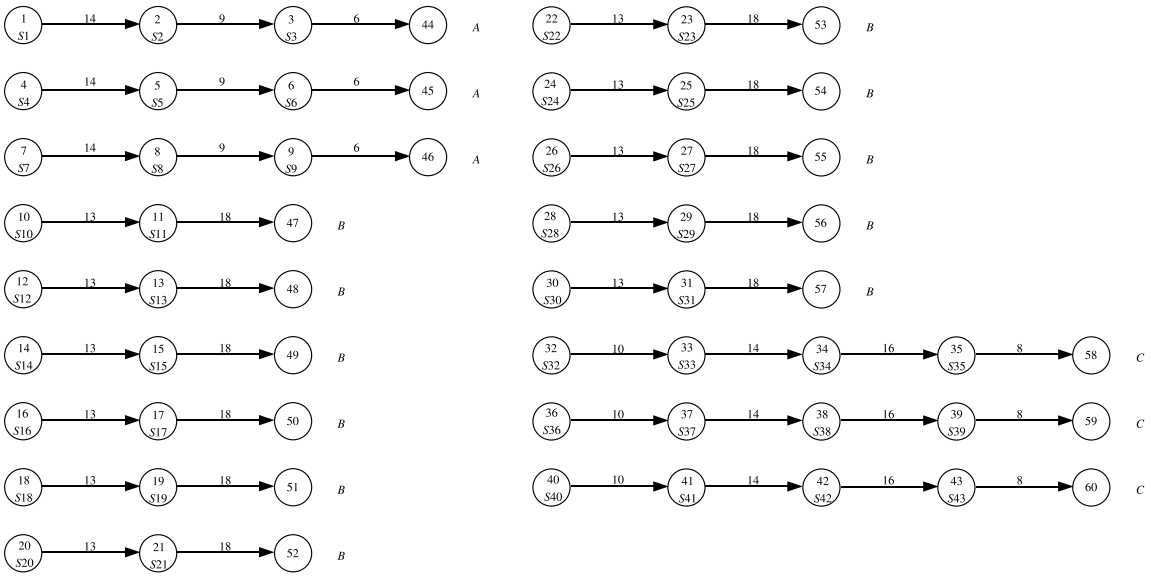
Taszk	A termék		B termék		C termék	
	Berendezés	Idő (h)	Berendezés	Idő (h)	Berendezés	Idő (h)
1	$E1$	14	$E2$	13	$E2$	10
			$E3$	13	$E3$	10
2	$E2$	9	$E1$	18	$E4$	14
	$E3$	9			$E5$	14
	$E4$	11				
	$E5$	11				
3	$E4$	6			$E2$	16
	$E5$	6			$E3$	16
4					$E4$	8
					$E5$	8

4.7. feladat

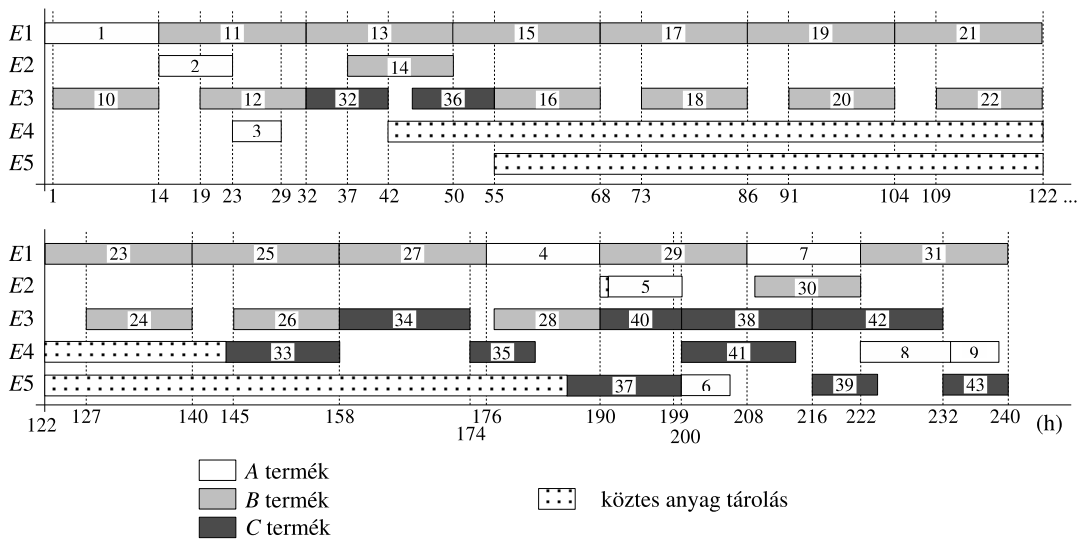
A 4.7. feladat receptje a 4.9 táblázatban található. A feladatban az A termékből 3, a B termékből 11, a C termékből 3 batchnyi mennyiséget kell ütemezni. A recept-gráf a 4.21 ábrán található. A recept-gráfban az $S1 = S4 = S7 = \{E1\}$, $S2 = S5 = S8 = \{E2, E3, E4, E5\}$, $S3 = S6 = S9 = \{E4, E5\}$, $S10 = S12 = S14 = S16 = S18 = S20 = S22 = S24 = S26 = S28 = S30 = \{E2, E3\}$, $S11 = S13 = S15 = S17 = S19 = S21 = S23 = S25 = S27 = S29 = S31 = \{E1\}$, $S32 = S36 = S40 = \{E2, E3\}$, $S33 = S37 = S41 = \{E4, E5\}$, $S34 = S38 = S42 = \{E2, E3\}$ és $S35 = S39 = S43 = \{E4, E5\}$.

A feladat optimális megoldásának végrehajtási ideje 240 óra. A feladat megoldásához az EQ-SG algoritmussal 67 másodpercre volt szükség, a TA-SG algoritmus 12 óra futási idő alatt a feladatot nem oldotta meg[5]. Az optimális megoldás Gantt-diagrammja a 4.22 ábrán található.

[5] Az algoritmus egy 1,5 GByte DDR RAM-mal felszerelt, Intel Pentium 4 3.06GHz órajelű PC-n futott.



4.21. ábra. A 4.7. feladat recept-gráfja.



4.22. ábra. A 4.7. feladat egy optimális ütemezésének Gantt-diagrammja.

Véletlen ütemezési feladatok megoldása az EQ-SG és a TA-SG algoritmusokkal

Ebben a részben „átlagos" ütemezési feladatok megoldásával és a futási eredmények összehasonlításával vizsgálom az EQ-SG és a TA-SG algoritmusok hatékonyságát. Az összehasonlításhoz véletlen ütemezési feladatokat állítottam elő. Az ütemezési feladatokban a recept strukturája és a felhasználható berendezések száma rögzített, az ütemezési feladat további paramétereinek meghatározására véletlenszám generátort használok. A generátor egyenletes eloszlású egész számokat szolgáltat adott intervallumban.

A receptben A, B és C terméket kell előállítani $E1$, $E2$, ..., $E8$ berendezések ütemezésével. A receptben az A terméket három, a B terméket kettő, a C terméket pedig négy egymás utáni taszk végrehajtásával lehet legyártani. A termékekből előállítandó mennyiségeket véletlenszám generátor az $[1, 4]$ zárt intervallumból választja ki. A véletlenszám generátor a recept-gráf minden taszkjához egy, vagy két berendezést választ ki. A berendezések végrehajtási ideje az $[5, 80]$ zárt intervallumból egy egész érték.

A véletlen ütemezési feladatok jellemzésére meghatároztam a berendezések „terheltségét". A berendezés terheltségét a hozzá rendelhető taszkok végrehajtási idejeinek összege határozza meg oly módon, ha egy taszkot több berendezés is végrehajthat, akkor az a taszk a végrehajtási idő berendezésre eső hányadával vesz részt a terheltségben. Az ütemezési feladatot jellemzi a berendezések terheltségeinek az átlaga és szórása. A terheltség szórása és átlaga megadja az ütemezési feladat relatív szórását. A relatív szórás jellemzi az átlagtól való eltérés százalékos nagyságát. A relatív szórás összehasonlíthatóvá teszi a különböző méretű ütemezési feladatokat.

Ötszáz véletlen feladat megoldásának futási idejeit és relatív szórását gyűjtöttem össze. A vizsgált véletlen feladatok relatív szórása $0,41$ és $1,32$ közötti értékek voltak. A véletlen feladatok felénél a terheltség relatív szórása kisebb mint 0.91. Ezeknél a feladatoknál 51 esetben az EQ-SG algoritmus 199 esetben az TA-SG algoritmus oldotta meg gyorsabban az ütemezési feladatot. Ha a terheltség relatív szórása nem kisebb, mint 0.91, akkor 97 esetben az EQ-SG és 143 esetben a TA-SG algoritmus

volt hatékonyabb.

A mérési eredmények igazolják, hogy kiegyenlített feladatok esetén (terheltség relatív szórása kicsi) a TA-SG algoritmus az EQ-SG algoritmushoz képest jobb eredményeket ér el ahhoz képest, mint amikor az ütemezési feladat túlterhelt berendezéseket is tartalmaz (terheltség relatív szórása nagy). Az ütemezési feladat strukturájának és a véletlenszám generátor paramétereinek hangolásával az EQ-SG és TA-SG algoritmus hatékonyságának aránya változtatható.

4.5. Összefoglalás

Szakaszos üzemű berendezések ütemezése összetett, kombinatorikus jellegű feladat. Az S-gráf módszertan [33, 84, 86] alapján kifejlesztett EQ-SG algoritmus a feladatosztály kombinatorikus tulajdonságát kihasználva ipari méretű feladatok megoldását teszi lehetővé.

Bevezettem és bemutattam egy S-gráf alapú új döntési stratégiát használó szétválasztási eljárást. Az új eljárás alapján implementáltam a TA-SG algoritmust. A TA-SG algoritmus biztosítja az optimális ütemezés megtalálását, ugyanakkor az EQ-SG algoritmushoz képest további kedvező tulajdonságokkal rendelkezik.

Ha a feladat tartalmaz egy olyan berendezést, mely a többihez képest túlságosan le van terhelve és tulajdonképpen ennek a berendezésnek az ütemezése határozza meg az ütemezés végrehajtási idejét, akkor az EQ-SG algoritmus hatékonyabban megtalálja a feladat optimumát. Ekkor a leterhelt berendezés optimális ütemezésével a feladat végrehajtási ideje adva van. A többi berendezés ütemezése már könnyen megadható ezek után. A TA-SG ütemező akkor bizonyult hatékonyabbnak, mint az EQ-SG algoritmus, ha a feladat megoldásában a berendezések egyenletesen vannak leterhelve, nincsen olyan berendezés, melyre a többihez képest sokkal több taszk van ütemezve.

5. fejezet

Szakaszos működésű termelő folyamat korlátozott tisztítási költségű ütemezése

Szakaszos üzemű berendezések rugalmasságuk miatt széles körben használt eszközök műszaki gyártási folyamatokban. A festékgyártó iparban is szakaszos berendezéseket használnak, mely berendezések optimális, vagy közel optimális ütemezése kulcsfontosságú kérdés. Mivel a gyártási folyamatban a berendezések tisztítása egy olyan költséges művelet, amely közben nagy mennyiségű szennyezőanyag keletkezik, ezért ipari, gazdasági és környezetvédelmi elvárás olyan ütemezések meghatározása, melyek „elfogadható" tisztítási költséggel járnak (*waste minimization*), ugyanakkor az ütemezés végrehajtási ideje minimális.

A 3. fejezetben ismertetett S-gráf módszertan hatékony és rugalmas eszköz többcélú szakaszos folyamatok ütemezésére [84, 86]. Ebben a fejezetben az S-gráf módszertanban bevezetem a berendezések működése során fellépő ütemezési sorrend függő tisztítási műveleteket, a tisztítási műveletek idő és költség vonzatát. Bemutatok egy algoritmust a korlátozott tisztítási költséggel járó minimális végrehajtási idejű ütemezés meghatározására. Az algoritmus hatékonyságát és működését ipari festékgyártási feladat megoldásával szemléltetem.

Az EQ-SG algoritmusnál ismertetett leghosszabb út kereső algoritmus helyett egy

lineáris programozási modellt használok a részfeladatok alsó korlátjának meghatározására [33]. A modellt használva élesebb alsó korlátot kaphatunk a részfeladat végrehajtási idejére a keresési fa gyökérnek közelében.

5.1. Berendezések tisztítását figyelembe vevő ütemezési módszerek szakirodalma

A festékgyártó ipar méreteit és jelentőségét nagyon jól jelzi, hogy egyedül az Amerikai Egyesült Államok területén belül több mint ezer vállalkozás foglalkozik festék, vagy alapozó anyagok előállításával. Ezek a vállalkozások nagy mennyiségben állítanak elő festékeket, melyeket változatos területeken használhatnak fel. Az itt készült festékeket és alapozókat arra használják, hogy megvédjék, kezeljék és szépítsék a tárgyak, épületek felületét. A festékek legjelentősebb felhasználási területei az építőipar (pl. épületek festése), az ipari festék (pl. autógyártás, fa felületek kezelése), vagy a speciális felületek festése, kezelése (pl. közlekedési jelek festése, tetőzet kezelése).

A festék gyártása általában három fő lépésben történik. Először a pigmentek örlése és diszpergálása történik, ezt követi a keverés művelete, majd végezetül a kész festék csomagolása [65]. A festékek és alapozók előállítása általában szakaszos berendezésekben történik. A gyártás során fix és mozgatható berendezéseket is használnak, mint például nagy sebességű diszperziós keverőket (*high-speed dispersion mixer*), forgó szakaszos működésű keverőket (*rotary batch mixer*), örlő malmokat (*sand mill*), vagy tárolókat (*tank*). Nyersanyagként oldószereket, gyantát, pigmenteket és egyéb organikus vagy nem organikus adalékokat használnak. A festékgyártás folyamán általában nem megy végbe a nyersanyagok között kémiai reakció, hanem a festék gyártása egyedül a nyers- és köztes anyagok megfelelő arányú keverését jelenti. Mivel egy festék, vagy alapozó anyag gyártó üzem nagy mennyiségű különböző fajtájú festékeket állít elő, ezért egy ilyen üzem optimális ütemezésének meghatározása nehéz, nagy bonyolultságú feladat. A festékgyártó ipar jelentőssége és nagy volumene miatt fontos az optimális, vagy optimális közeli ütemezések hatékony meghatározása.

A gyártás során általában berendezések tisztítása során keletkezik a legnagyobb

mennyiségben szennyezőanyag. Egy berendezést általában akkor kell tisztítani, mielőtt egy másik fajta termék előállítására kezdjük el használni. Ezért lehetőség szerint a termékek közötti váltások számát minimalizálni kell ahhoz, hogy környezetszennyezés szempontjából optimális, vagy jobb gyártási folyamatot kapjunk. Szakaszos és folytonos folyamatok berendezéseinek tisztítása jelentősen eltérhetnek egymástól. Markowski és Urbaniec folytonos folyamat berendezéseinek on-line tisztítására dolgoztak ki egy MINLP modellt [50]. A tisztítással és minél megbízhatóbb működéssel a folyamat összköltsége csökkenthető.

Az ipari feladatok jelentős részében a berendezések konfigurációs ideje (*setup time*) nem hanyagolható el és nem építhető be a taszkokra szánt végrehajtási időbe, mivel ez az idő az ütemezés sorrendjének a függvénye. Ilyen feladatokat találhatunk például a nyomdaiparban, textiliparban, vagy a vegyiparban.

A nyomdaiparból származó ütemezési feladatok során a beállítási és a végrehajtási idő különválik egymástól. Itt a nyomdai gépek beállítási (tisztítási) ideje függ a használt színek sorrendjétől. Hasonló sorrendfüggő tisztítás idők jelentkeznek a vegyiparban, gyógyszeriparban, élelmiszeriparban, vasiparban és a félvezetők gyártása során.

Az ütemezési feladatokkal foglalkozó kutatások nagy részében vagy elhanyagolják a berendezések beállításához szükséges időt, vagy feltételezik, hogy ez az idő a végrehajtott taszkok sorrendjétől nem függ. Tan és társai az egy berendezést tartalmazó a késéseket minimalizáló ütemezési feladatot vizsgálták [95]. A berendezések a taszkok sorrendjétől függő konfigurálási idővel rendelkeznek. A vizsgált ütemezési feladatot szétválasztás és korlátozás, genetikus, szimulált hűtés és véletlen-kezdéssel induló páronkénti csere elvén működő algoritmusokkal oldatták meg. A genetikus és szimulált hűtés algoritmusok nagy méretű feladatok megoldására ad lehetőséget, azonban a megoldás optimalitását nem garantálják. A szétválasztás és korlátozás elvű algoritmusuk megadja a feladat optimális megoldását, azonban a megoldott feladat mérete kisebb. Általában akkor kell nagy berendezés konfigurálási idővel számolni két végrehajtott taszk között, ha a taszkok szignifikánsan különböző erőforrásokat, vagy eltérő gyártási technológiát használnak. Parthasarathy és Rajendran flow shop feladat taszk ütemezési sorrend függő konfigurálási idővel rendelkező berendezések

ütemezésére szimulált hűtés alapú algoritmust használnak [68].

Az ütemezési feladat matematikai modellje nem tartalmazhatja az összes létező és jelentős mérnöki szempontot, hiszen akkor egy olyan modellt kellene megoldanunk, melyre kevés az esély. Ehelyett hatékonyabb megközelítés, ha meghatározzuk egy egyszerűbb modell több optimális, vagy optimális közeli megoldását és a későbbiek során ezeket a megoldásokat elemezve kiválasztjuk azt, amelyik legjobban megfelel a teljesíteni kívánt szempontjainknak. A kiválasztott megoldás azonban az esetek nagy részében nem lehet optimális.

Orcun és társai létrehoztak az STN modell alapján [41] egy olyan MILP modellt [64, 65], mellyel szakaszos tervezési és ütemezési feladatait modellezhetjük a festékgyártásnak. Egy olyan általános üzemet modelleznek, melyben a termékek különböző állomásokon, gyártási úton haladhatnak át. Különböző termékek különböző állomásokon készülnek. Az STN ábrázolás alapján felírt MILP modell alkalmas batch megosztás, berendezés karbantartás és határidők kezelésére is.

Méndez és társai véges erőforrással rendelkező szakaszos flow shop feladatok ütemezésére MILP modellt dolgoztak ki [54, 51]. A modell az STN alapú folytonos időábrázoláson alapul, tartalmazza az ütemezési sorrendtől függő berendezés konfigurálási időket. Méndez és Cerda az előző modell egy újszerű folytonos MILP modellé alakították át, mely a feladat optimális ütemezését adja [52]. A modellben különböző köztes anyag tárolási stratégiákat kezelnek (pl. UIS, NIS).

Rajendran és Ziegler flow shop ütemezési feladatok megoldására dolgoztak ki heurisztikus algoritmust, ahol a berendezések konfigurálásához, beállításához szükséges idő a végrehajtott taszkok sorrendjétől függ [75, 76]. A vizsgált feladatban a célfüggvény a feladatok befejezési idejének súlyozott összegének minimuma.

Az egyre szigorúbb környezetvédelmi előírások miatt gondosan meg kell választani milyen technológiákat alkalmazunk a berendezések tisztítására, illetve hogyan lehet csökkenteni a szükséges tisztítások számát. Grau és társai bevezettek egy modellt, melyben a tisztítás káros anyag kibocsátása függ a berendezés ütemezési sorrendjétől [26]. A kidolgozott algoritmus visszalépéses (*backtracking*) jellegű.

Rabadi és társai az egy berendezéses ütemezési feladat határidőhöz közeli ütemezésére szétválasztás és korlátozás típusú algoritmust fejlesztettek ki [74]. A feladatban

figyelembe vették a taszkok sorrendjétől függő berendezéskonfigurálási időt. Olyan ütemezést keresnek, melyben a taszkok befejezési idejében mutatkozó késések és sietések (koraiságok) összege minimális egy közös határidőhöz képest. Az ilyen típusú ütemezési feladatokat JIT ütemezési feladatoknak nevezik a szakirodalomban.

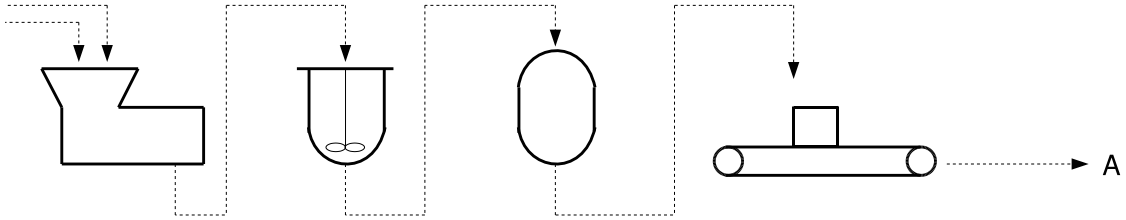
A háromgépes flow shop ütemezési feladatot oldották meg Allahverdi és Al-Anzi a szétválasztás és korlátozás módszerével [1]. A feladatban a tevékenységek beállításához (konfigurálásához) és végrehajtásához szükséges időt külön kezelik. A feladat célfüggvénye az ütemezés minimális végrehajtási ideje. Heurisztikák segítségével felső korlátot származtatnak, mellyel gyorsítják a feladatok megoldásának menetét.

Danneberg és társai flow shop feladatokra kidolgozott heurisztikát használó ütemezési algoritmusokat hasonlítottak össze [17]. Az ütemezési feladatban a taszkokat csoportokba rendezik. Egy csoport taszkjait egyszerre hajtják végre ugyanazon a berendezésen. A berendezés konfigurálását a csoport elkezdése előtt kell végrehajtani, a konfiguráláshoz szükséges idő a csoport taszkjaitól függ.

Tahar és társai a taszkok sorrendjétől függő berendezés konfigurálási időket tartalmazó ütemezési feladatot oldottak meg munkájukban [94]. A feladatban egyforma gépek minimális végrehajtási idejű ütemezését keresték. Az ilyen feladatok NP nehezek. A megoldást heurisztikus algoritmus segítségéve határozzák meg.

Choi és Choi alternatív recepttel és taszk sorrend függő konfigurálási idővel működő berendezéseket tartalmazó job shop ütemezési feladat megoldásával foglalkoztak munkájukban [14]. A feladat megoldására MIP (*Mixed Integer Programming*) matematikai programozási modellt írtak fel, majd lokális keresést használó algoritmusokkal oldották meg a feladatot.

Zdrałka az egy berendezéses batch ütemezési feladat megoldására adott heurisztikus algoritmust munkájában [101]. A feladatban minden jobhoz tartozik egy végrehajtási idő és egy szállítási idő. A gép a jobokat batchekbe rendezve hajtja végre. Ha különböző batchhez tartozó jobokat hajt végre a gép egymás után, akkor a gép beállításához időre van szükség. A feladat a végrehajtási idő minimalizálása. Zdrałka heurisztikus algoritmust ismertet, mely legrosszabb esetben az optimumhoz képest 3/2-es ütemezést szolgáltat. A kritikus út a jobok olyan sorozata, mely az ütemezés végrehajtási idejét meghatározza. Az úgynevezett „becslő algoritmus” a kritikus út



5.1. ábra. Festékgyártást leíró recept.

módosításával dolgozik, azaz megpróbál a kritikus útból eltávolítani jobokat azért, hogy csökkenjen a végrehajtási idő.

5.2. A megoldandó festékipari feladat

A termékek gyártási folyamatát leíró receptek megadhatóak egy-egy taszk-hálózat formájában (részletesen lásd a 3. fejezetben). Ebben a taszk hálózatban a festékek gyártásához négy egymást követő taszkot kell végrehajtani. Ezek a taszkok a következőek: örlés, keverés, tárolás és csomagolás. Az 5.1 ábra a festék gyártását leíró receptet ábrázolja. A receptben az örlés, keverés és tárolás műveletet szakaszos üzemű berendezésekkel, míg a csomagolást folytonos működésű berendezésekkel lehet végrehajtani. A gyártás során biztosítani kell, hogy a keverés művelet után a termék adott ideig egy tároló berendezésben várakozik, mielőtt a termék csomagolása elkezdődhet. Ez a várakozás a festék buborékoltatása miatt szükséges. Ez a várakozási idő a tároló berendezések működési idejében lesz figyelembe véve.

Mivel berendezésből kevesebb van, mint ahány taszkot végre kell hajtani, ezért nem rendelhetünk hozzá mindegyik taszkhoz egy önálló berendezést, a berendezéseket ütemezni kell. Az ütemezés során minden taszkhoz hozzá kell rendelni egy időintervallumra egy megfelelő berendezést, mely intervallum alatt a berendezés a taszkot végrehajtja (az intervallum hossza legalább a taszk működési idejének hossza). Ha az ütemezés alapján a berendezés két olyan taszkot hajt végre egymás után, mely taszkok között a berendezést tisztítani kell, akkor a tisztításhoz szükséges időt a berendezés váltási idejében ábrázoljuk. A festékgyártás során a örlő, keverő és tároló

berendezések tisztítását kell figyelembe vennünk. A gyártási folyamatban keletkező összes köztes anyagot felhasználják a recept alapján rákövetkező taszkok.

Ütemezési feladatokban általában a legkisebb végrehajtási idejű (*makespan*) ütemezés meghatározása a cél. Az ilyen ütemezés a legtermelékenyebb, azonban feleslegesen nagy számú tisztítást és ezzel együtt szennyező anyag kibocsátást von magával. Azért, hogy figyelembe tudjam venni az S-gráf módszertanban az ütemezéshez kapcsolódó tisztítási költséget, a korlátozás eljárásban a végrehajtási idő mellett a tisztítási költséget is számolni kell. Az eredeti S-gráf módszertan célfüggvénye a minimális végrehajtási idő, a festékgyártási feladatban a célfüggvény nem változik, a tisztítási költség felső korlátja az ütemezés megvalósíthatóságát befolyásolja. Egy új korlátozási szempont jelentősen nem befolyásolja az S-gráf módszertan alapalgoritmusait, ezért az eredeti S-gráf módszertan hatékonyan alkalmazható a festékipari feladat megoldására is.

Az elemzett és megoldott ipari feladatban a csomagolók a legkihasználtabb berendezések, az örlő és keverő berendezések átlagosan kihasználtak, míg a tárolók a legkevésbé kihasználtak. Ezt az információt a szétválasztás döntési sorrendjében kihasználva gyorsabb, hatékonyabb megoldót kapunk.

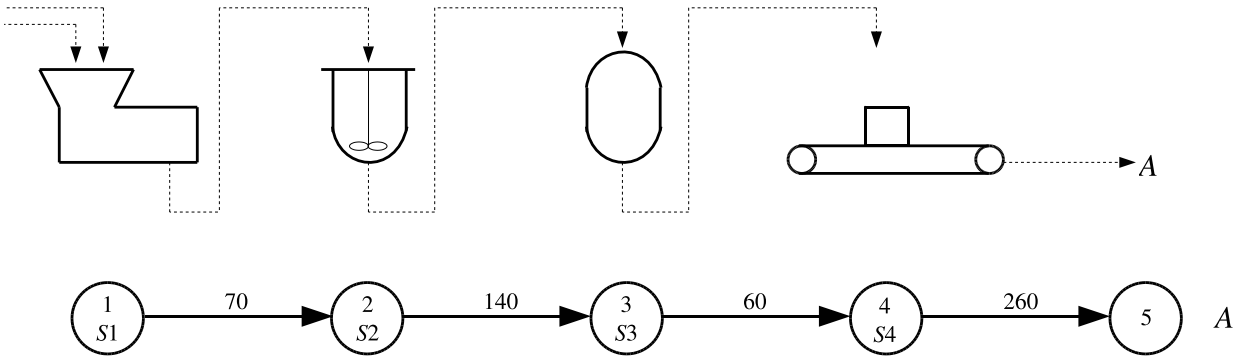
5.3. S-gráf módszertan alkalmazása a festékgyártási feladatra

A 3. fejezetben bemutatott EQ-SG algoritmus gyorsítási eszközökkel kiegészítve hatékony algoritmus lehet nagyméretű ipari ütemezési feladatok megoldására. A festékgyártási feladat speciális tulajdonságai módosítják az alap megoldót.

A festék gyártás receptje a korábban ismertetett úton átalakítható recept-gráffá. Az 5.2 ábrán a recept és a receptből származtatott recept-gráf látható.

5.3.1. Keresési tér csökkentése

Az ipari ütemezési feladatokban nagy számú batchben kell ugyanazokat a termékeket előállítani, ütemezni. Az S-gráf alapalagoritmus minden előállított batcht, még ha



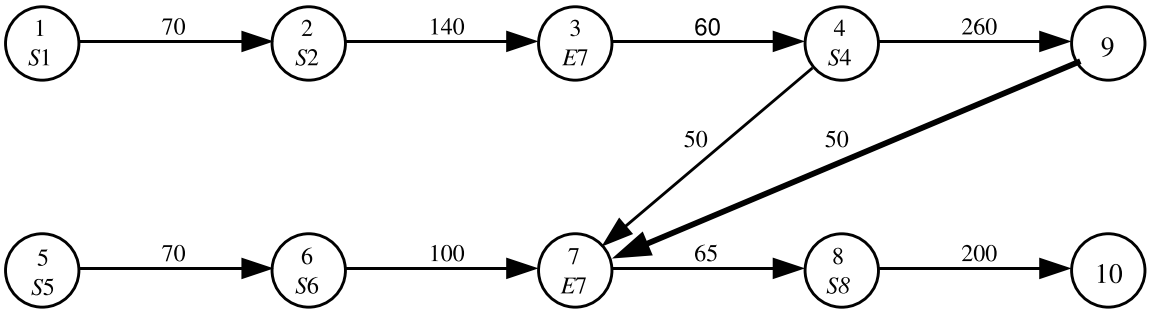
5.2. ábra. Recept-gráf készítése a festékgyártás receptjéből.

ugyanahhoz a termékhez is tartozik külön termékként kezeli és így feleslegesen sok redundáns megoldást tartalmaz a keresési tér. Ezeknek a redundáns megoldásoknak a kizárására Holczinger és társai bevezették a segéd-éleket [34], mint további korlátozásokat a recept-gráfba, mellyel jelentős gyorsítást ért el az algoritmus futása során, miközben az optimális megoldást továbbra is megkapta. A segéd-élekkel nagy ipari ütemezési feladatok megoldására nyílt lehetőség. A keresési tér ily módon való csökkentésével a festékgyártási feladat ipari alkalmazására nyílik lehetőség. A segéd-él technika részletes ismertetése a 3. fejezetben található.

5.3.2. Tároló berendezések ütemezése

A vizsgált ipari feladatban az NIS tárolási stratégiát kell alkalmazni, azaz a köztes anyagokat mindvégig valamilyen erre alkalmas berendezésben tárolni kell. A rendszerben a köztes anyagokat az örlő, keverő és tároló berendezésekben lehet tárolni. A csomagolók nem tudják az éppen csomagolás alatt álló festék mennyiséget tárolni, hanem egy tároló berendezésnek kell rendelkezésre állnia addig, míg az összes festéket be nem csomagoljuk. Az ütemezésnek biztosítania kell, hogy a köztes anyagot tároló berendezések a csomagolók működése alatt nem használhatóak új taszkok végrehajtására.

Ez a tárolási korlát könnyen kifejezhető egy új, módosított ütemezési-él bevezetésével. Például az 5.3. ábrán levő S-gráf 9-es és 7-es csúcsok közötti ütemezési-éle



5.3. ábra. A 9-es és 7-es csúcs közötti él biztosítja, hogy az $E7$ berendezés, csak a 4-es taszk (csomagolás típusú) befejezése után kezdheti meg a 7-es taszkot.

biztosítja azt, hogy a köztes festéket az $E7$-es tárolóban tároljuk miközben a csomagolás be nem fejeződik. Vegyük észre, hogy az 5.3. ábrán a 4-es és 7-es csúcs közötti eredeti ütemezési-él redundáns, elhagyható.

5.3.3. Lineáris programozási modell a részfeladat végrehajtási idejére érvényes alsó korlát meghatározására

A leghosszabb út kereső algoritmus nem szolgáltat eléggé éles alsó korlátot a keresési fa gyökeréhez közeli részproblémákra. A leghosszabb út kereső algoritmus akkor hatékony, ha a recept-gráf elég sok ütemezési-élt tartalmaz. Lineáris programozási modell felírásával és megoldásával a leghosszabb út kereső algoritmusnál nem kisebb, élesebb alsó korlátot kaphatunk.

A korlátozás eljárás során használt lineáris programozási modellt Holczinger Tibor disszertációjában ismertette [33]. A lineáris programozási modellt a D. függelék tartalmazza.

5.4. Algoritmus a festékgyártási feladatra

A festékgyártási feladat megoldására a korábban ismertetett EQ-SG algoritmus (3.5-3.7 ábrák) alapján dolgoztam ki a P-SG algoritmust. A P-SG algoritmus eljárásait az EQ-SG algoritmus *EQ-SG*, *EQ-szétválasztás* és *EQ-korlátozás* eljárásához hasonlóan

P-SG, *P-szétválasztás* és *P-korlátozás* eljárásnak nevezem.

A P-SG algoritmus *P-SG* eljárásának a bemenete a festékgyártási feladat, ami a recept-gráfot és a tisztítási költség korlátját tartalmazza. Jelöljük C_{max}-szal a tisztítási költségek korlátját. A *P-SG* eljárás az *EQ-SG* eljárás alapján inicializálja az adatstruktúrákat és kezeli a nyitott részfeladatok halmazát (SET).

A *P-szétválasztás* eljárás az *EQ-szétválasztás* eljárás lépéseit követi, figyelembe véve a tároló típusú berendezések feladatspecifikus ütemezését. Az eljárás a 5.3.2 fejezetben ismertetett módon kezeli a tárolók ütemezési-éleit.

A *P-korlátozás* eljárást a 5.3.3 fejezet és a tisztítási költség korlát alapján újraépítettem az *EQ-korlátozás* eljárást (lásd. 5.4 ábra). A *P-korlátozás* eljárás a körkereső algoritmussal ellenőrzi, hogy a részfeladat megvalósítható ütemezés-e. Ha megvalósítható, akkor a részfeladathoz tartozó részütemezés ütemezési-éleihez tartozó tisztítási költségeket összegzi. Ha az összeg nem nagyobb, mint a tisztítások felső korlátja, akkor az LP modellt alkalmazva határozza meg a részfeladat végrehajtási idejének alsó korlátját. Ha a körkereső algoritmus kört talált, vagy a tisztítási költségek összege nagyobb, mint a tisztítási költség korlátja, akkor a részfeladat alsó korlátja végtelen, tehát nem kaphatunk belőle megoldást a feladatra.

5.5. Ipari feladat

Tegyük fel, hogy hat terméket (A, B, C, D, E és F) kell előállítani a rendelkezésre álló huszonhárom berendezéssel ($E1$-$E23$). A termékeket gyártását leíró receptet az 5.1 táblázatban találhatjuk.

Legyen az $E6$-$E9$ berendezések váltási ideje 70 perc, az $E1$-$E5$ és $E10$-$E20$ berendezések váltási ideje pedig 100 perc, ha egy új terméket kezdenek el gyártani. Minden más váltási idő értéke legyen 0. A termékenként előállítandó batchek számát az 5.2 táblázat tartalmazza.

A feladat minimális végrehajtási idejű megoldásának értéke 6700 perc. A minimális végrehajtási idejű megoldás meghatározásához létrehoztunk egy MILP modellt Méndez és társainak munkája alapján [52]. A modellt GAMS/CPLEX 7.5 általános

jelölések:
C_{max}: feladat tisztítási költségének korlátja
$cost_{(e,i,j)}$: az e berendezés i és j taszk közötti tisztítási költsége
procedure *P-korlátozás*(PP)
begin
kör_kereső(G(PP));
if no *cycle*
begin
if $\sum cost_{(e,i,j)} \leq C_{max}$, ahol (i,j) az e berendezés ütemezési-éle G(PP)-ben
bound :=*lpmodell*(G(PP));
else
bound := ∞;
end
else
bound := ∞;
return *bound*;
end.

5.4. ábra. A *P-korlátozás* eljárás.

5.1. táblázat. Ipari feladat receptje.

Taszk	A termék		B termék		C termék	
	Berendezés	Idő (perc)	Berendezés	Idő (perc)	Berendezés	Idő (perc)
1	$E1$	60	$E1$	60	$E2$	60
2	$E6$	310	$E7$	240	$E8$	120
			$E8$	120		
			$E9$	240		
3	$E10$	60	$E11$	120	$E11$	120
	$E11$	120	$E13$	60	$E12$	70
	$E13$	60	$E15$	120	$E13$	70
	$E15$	120	$E17$	60	$E14$	60
	$E17$	60	$E19$	120	$E16$	50
	$E19$	120	$E20$	60		
	$E20$	60				
4	$E21$	720	$E22$	540	$E21$	720
	$E22$	540	$E23$	720	$E22$	540

Taszk	D termék		E termék		F termék	
	Berendezés	Idő (perc)	Berendezés	Idő (perc)	Berendezés	Idő (perc)
1	$E3$	60	$E4$	40	$E5$	40
2	$E7$	240	$E6$	300	$E7$	240
	$E9$	240	$E8$	120	$E8$	120
3	$E10$	60	$E10$	60	$E10$	60
	$E11$	120	$E12$	90	$E15$	120
	$E13$	60	$E14$	90	$E16$	90
	$E14$	90	$E16$	90	$E17$	60
	$E15$	120	$E18$	90	$E18$	90
	$E17$	60	$E20$	60	$E19$	120
	$E18$	90			$E20$	60
	$E19$	120				
	$E20$	60				
4	$E22$	540	$E21$	720	$E21$	720
	$E23$	720			$E22$	540
					$E23$	720

5.2. táblázat. Batch-ek száma termékenként.

Termék	A	B	C	D	E	F
Batch-ek száma	3	5	1	3	9	3

5.3. táblázat. Örlő berendezések ($E1$-$E5$ berendezések) tisztítási költségei (CU).

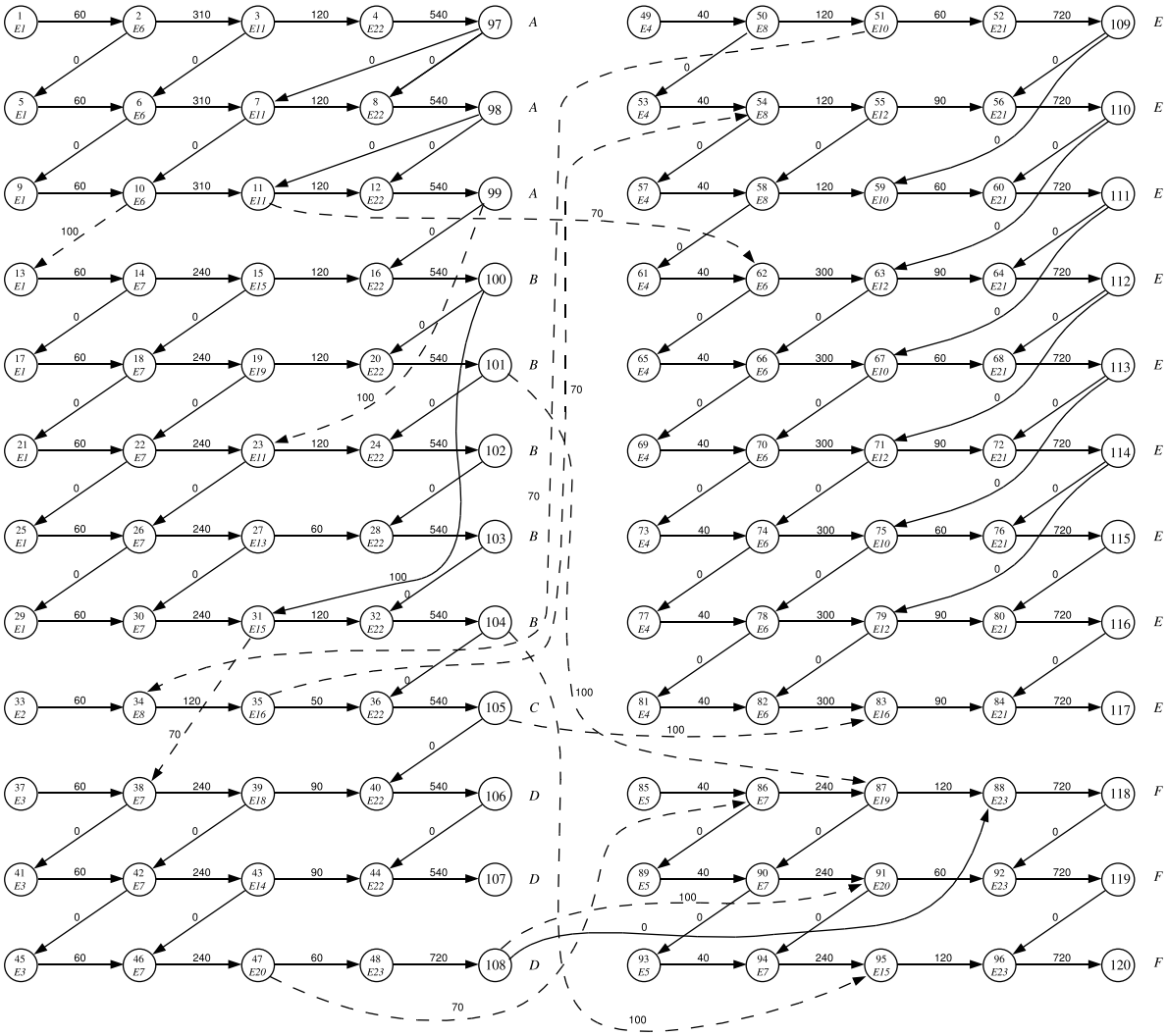
		Hova					
		A	B	C	D	E	F
Honnan	A	0	500	500	500	500	500
	B	1000	0	500	500	500	500
	C	1000	500	0	500	500	500
	D	1000	500	500	0	500	500
	E	2000	1000	1000	1000	0	500
	F	2000	1000	1000	1000	500	0

célú megoldóval 34 másodperc alatt sikerült megoldani[1]. Ugyanezen a PC-n az EQ-SG algoritmus 0,9 másodperc alatt határozta meg a feladat optimális megoldását. Az 5.5 ábra mutatja a feladat egyik optimális (tehát minimális) végrehajtási idejű megoldásának ütemezési-gráfját. Az ütemezési-gráfban szaggatott élek jelölik azokat a berendezés váltásokat, melyek tisztítással járnak, mert a berendezés egy új termék előállítását kezdi el.

A berendezések tisztítása az egyik legköltségesebb és legszennyezőbb művelet a gyártási folyamatban. Az 5.5 ábrán bemutatott minimális végrehajtási idejű ütemezés során 11-szer kell valamely berendezés tisztítását elvégezni. Az ütemezés összes tisztítási költségét a szaggatott ütemezési-élek alapján számítható. A tisztítási művelet bonyolultsága alapján megadhatóak a tisztítások költségei (*Cost Unit*, CU). Az örlő berendezések tisztítási költségét az 5.3 táblázat, a keverőkét az 5.4 táblázat és a tárolókét az 5.5 táblázat tartalmazza.

Az 5.3-5.5 táblázatok alapján a 11 tisztítást tartalmazó minimális végrehajtási

[1] Az algoritmus egy 1 GByte DDR RAM-mal felszerelt, AMD Athlon XP 2200 MHz órajelű PC-n futott.



5.5. ábra. Minimális végrehajtási idejű ütemezési-gráf, melyben a szaggatott élek tisztítási költséggel járó váltásokat jelölnek.

5.4. táblázat. Keverő berendezések ($E6$-$E9$ berendezések) tisztítási költségei (CU).

		Hova					
		A	B	C	D	E	F
Honnan	A	0	1500	1500	1500	1500	1500
	B	2500	0	1500	1500	1500	1500
	C	2500	1500	0	1500	1500	1500
	D	2500	1500	1500	0	1500	1500
	E	5000	2500	2500	2500	0	1500
	F	5000	2500	2500	2500	1500	0

5.5. táblázat. Tároló berendezések ($E10$-$E20$ berendezések) tisztítási költségei (CU).

		Hova					
		A	B	C	D	E	F
Honnan	A	0	1000	1000	1000	1000	1000
	B	2000	0	1000	1000	1000	1000
	C	2000	1000	0	1000	1000	1000
	D	2000	1000	1000	0	1000	1000
	E	4000	2000	2000	2000	0	1000
	F	4000	2000	2000	2000	1000	0

idejű ütemezés tisztítási költsége 14000 CU. Ugyanakkor a minimális tisztítási költségű ütemezés értéke 3500 CU és mindössze 4 berendezés tisztítást tartalmaz. Ennek a minimális költségű ütemezésnek a végrehajtási ideje 6910 perc. Az ütemezés végrehajtási idejének körülbelül 3%-os növekedésével a tisztítási költségek a negyedére csökkennek.

A tisztítási költség és a végrehajtási idő egymással konkuráló mennyiségek. Ha az egyik mennyiséget javítjuk, akkor a másik általában romlik. Ezért a pareto optimális megoldások megtalálása fontos feladat lehet. Ha az egyik mennyiséget korlátozzuk, miközben a másikat optimalizáljuk megkapjuk a pareto optimális megoldások halmazát. Ha például a tisztítási költségek felső korlátját 5500 CU-nak választjuk, akkor a minimális végrehajtási idejű ütemezés ideje 6700 perc.

5.6. Összefoglalás

Munkámban szakaszos folyamatok optimális ütemezésére dolgoztam ki a P-SG algoritmust, melyben figyelembe veszem a berendezések tisztításának idejét és költségét. Az algoritmust az EQ-SG algoritmusból kiindulva hoztam létre. A P-SG algoritmus kezeli az iparból származó festékgyártási feladat speciális tárolási tulajdonságait, biztosítja az NIS tárolási stratégiát és a tároló berendezések különleges ütemezését.

Bemutattam egy új korlátozási modellt az ütemezési részfeladatok alsó korlátjának számítására. A bemutatott LP modellel legalább olyan jó alsó korlátot kapunk, mint a leghosszabb út kereső algoritmussal. Leginkább a keresési fa gyökérhez közeli részfeladataira ad lényegesen jobb alsó korlátot. Az LP modell az EQ-SG algoritmusban is használható a korlátozás eljárásban.

A fejezet témájához kapcsolódó publikációim

Nemzetközi folyóirat cikk

Adonyi, R., G. Biros, T. Holczinger, and F. Friedler, Effective scheduling of a large-scale paint production system, Journal of Cleaner Production, 16 (2), 225-232 (2008).

Nemzetközi konferencia előadás

Adonyi, R., G. Biros, and F. Friedler, Effective Scheduling of a Large-Scale Paint Production System, PRES 2004 (7th Conference on Process Integration, Modelling and Optimization for Energy Saving and Pollution Reduction), Prague, Czech Republic, August 22-26, 2004.

6. fejezet

Ütemezés hőintegrációval

Szakaszos folyamatok összetettségük miatt nagy számban tartalmaznak olyan anyagáramokat, melyek hőmérsékletét változtatni kell. Az anyagáram hőmérsékletének változtatása energia befektetéssel jár. Gazdasági és környezetvédelmi szempontokból fontos, hogy olyan ütemezéseket keressünk, melyekben az ütemezés mellett az energia fogyasztást is figyelembe vesszük, lehetőség van a minél magasabb szintű hőintegrációra, azaz a rendszer megfelelő hőmérséklet változtatást igénylő anyagáramának hőcserélő berendezésben való összekapcsolásával a termelés költségeinek csökkentésére. Az ütemezési és a hőintegrációs feladat egymás mellett jelentkezik a vegyipari szakaszos gyártási folyamatok nagy részében.

Az ütemezési és hőintegrációs feladat megoldható szekvenciálisan, azaz egymás után. Így például először az ütemezési feladat optimális megoldását keressük meg, majd erre a megoldásra leszűkítve keressük az hőintegrációs feladat optimális megoldását, vagy fordított sorrendben. Mivel az egyik feladat megoldása befolyásolja a másik feladat megoldását, ezért a szekvenciális megközelítés nem vezethet mindkét feladat szempontjából optimális megoldáshoz. Például egy optimális ütemezés esetén könnyen előfordulhat, hogy az nagyon kevés lehetőséget biztosít a hőintegráció számára, így magas lesz a gyártás energiaköltsége. Ugyanakkor ha az optimumhoz képest kicsivel rosszabb ütemezéseket is megengedünk, akkor ezeknél az ütemezéseknél megnövekedhet a hőintegráció lehetősége és így csökken a szükséges energia mennyisége.

A hőintegrációt hőcserélő berendezéseket alkalmazva valósíthatjuk meg. A gyakorlatban a hőcserélő berendezések igen drágák, a tisztításuk nehéz és költséges, esetleg lehetetlen. A hőcserélő berendezések ütemezését és így a szakaszos berendezések ütemezését is befolyásolja hőcserélő berendezések tisztításának nehézsége. A szakaszos gyártási folyamat ütemezése befolyásolja a potenciális hőcseréket, a gyártás anyagáramainak időfüggő jelenléte miatt. Cél lehet olyan ütemezések meghatározása, mely maximális lehetőséget biztosít hőcserék megvalósítására, ugyanakkor a végrehajtási ideje is elfogadható.

6.1. Szakaszos folyamatok hőintegrációjának szakirodalma

A hőcserélő hálózatok szintézise (*Heat Exchanger Network Synthesis*, HENS) során a hőcserélő berendezések egy olyan hálózatának (*Heat Exchanger Network*, HEN) a tervezése a cél, mely optimálisan kielégíti a termelési rendszerben fellépő hűtési és fűtési igényeket. A hűtési és fűtési igények kielégíthetőek a rendszerbe kívülről behozott energiával (pl. hűtőtorony vagy forró gőz). A költségek csökkentése érdekében azonban a meglévő fűtési és hűtési igényeket, mint hőelvonási lehetőségeket és hőforrásokat kiaknázva, a felhasznált külső energia mennyisége és így a termelési rendszer költsége is csökkenthető. Az optimális HEN meghatározásához figyelembe kell venni a hőcserélő egységek költségeit. A HENS feladatok általában nagy méretű, összetett, kombinatorikus jellegű feladatok. A hőcserélő hálózat magas költsége miatt az optimális HEN megtalálása fontos feladat.

A hőcserélő hálózat egy olyan a termelő folyamatba integrált folytonos alrendszer melyben egyes folyamok hőmérsékletét növelni, másokét pedig csökkenteni kell. Az előbbieket hidegáramoknak (kezdeti hőmérsékletük alacsonyabb, mint a végső), az utóbbiakat melegáramoknak nevezzük. A meleg- és hidegáramokat közösen hőáramoknak nevezzük. Ha a hőáram fajhője (fajlagos hőkapacitása) c, akkor m tömegű anyag hőmérsékletének ΔT-vel való megváltoztatásához $\Delta Q = cm\Delta T$ energiára van szükség. Folytonos rendszer esetén a képletben a tömeg helyett a folyam nagyságát

kell venni, az energia helyett pedig az időegység alatt befektetett energia nagyságát kapjuk.

A hőcsere működését a termodinamika első és második törvénye írja le. A termodinamika első törvénye az energia megmaradását mondja ki, azaz egy zárt rendszerben jelen levő energia összege nem változik. A termodinamika második törvénye azt mondja, hogy a hőáramlás iránya a melegebb helyről a hidegebb hely felé mutat. A hőcserélő hálózatok tervezésénél a második törvény alapján a cél a hőáramok többszöri felhasználása egészen addig, míg a törvény ezt megengedi.

Melegenergia szolgáltatónak (*hot utility*) azokat a gyártórendszeren kívüli eszközöket nevezzük melyek hőt juttatnak a rendszerbe a megfelelő hőigények kielégítésére. A hidegenergia szolgáltató (*cold utility*) a gyártási rendszerben jelen levő, fel nem használható hőt vonja ki. Melegenergia szolgáltatóként használhatunk gőzt, melyet bojlerekből, vagy megcsapolt turbinákból nyerhetünk. Hidegenergia szolgáltatóként vizet, vagy hideg levegőt használhatunk.

A melegáramokat és a melegenergia szolgáltatókat hőforrásoknak, a hidegáramokat és hidegenergia szolgáltatókat hőnyelőknek hívjuk. Egy hőforrás és hőnyelő között a hőátadás hőcserélő berendezésen keresztül történik. A hőcserélő berendezések lehetnek egyenáramúak, vagy keresztáramúak a bennük folyó anyagáramok irányai alapján. A fizikai kialakításuk alapján lehetnek kötegesek, lemezesek. A termodinamika második főtétele alapján akkor lehetséges hőcsere, ha a hőforrás melegebb, mint a hőnyelő, azonban a gyakorlatban gazdaságossági szempontokból megköveteljük, hogy egy minimális, ΔT_{min} hőmérséklet különbség legyen a hőforrás és a hőnyelő között.

Linnhoff és Flower munkájukban összefoglalták a hőcserélő hálózatok szintézisére bevezetett eljárásokat, gyakorlati alkalmazhatóságuk szempontjából [46]. Megállapították, hogy habár számos eljárást tettek közzé a szakirodalomban, azok nem, vagy nehezen alkalmazhatóak az iparban. A hőcserélő hálózatok tervezéséhez bevezették a pinch technológiát [47]. A pinch technológia egy grafikus eszköz, melyben a kaszkád diagrammon ábrázoljuk a rendszer összes hőáramát. A kaszkád diagramm a hőmérséklet – entalpia diagrammon ábrázolja a meleg és hideg kompozit görbéket, és megadja a rendszer hideg- illetve melegenergia igényét és pinch hőmérsékletét. A

pinch vagy szűkületi hőmérséklet olyan hőmérséklet értéket jelent, amelyen keresztül nem történik hőcsere egy minimális energia felhasználású hőcserélő hálózatban a pinch törvény alapján. A nagy kompozit görbe segítségével a tervező meghatározhatja milyen szolgáltatókat használjon a hőcserélő hálózatban. A pinch technológiával kapott megoldás hőcserélő hálózat általában 5%-nál nem rosszabb a rendszer globális optimumánál. Smith összefoglalta a hőcserélő hálózatok tervezésével kapcsolatos módszereket munkájában [90].

Rév és Fonyó, illetve Mizsey és Rév hőcserélő hálózatok tervezésére az eredeti pinch technológia helyett a pinch technológia egy általánosítását használják (*diverse pinch concept*) [57, 79]. A módszerrel már a tervezési fázisban figyelembe tudják venni az eltérő hőátadási tényezőket, így olyan csapdákat is el tudnak kerülni, melyek korábban csak a megvalósításnál jelentkeztek.

A pinch technológia mellett a szállítási modelleket is széles körben használják hőcserélő hálózatok szintézisére. A szállítási modellek a kaszkád diagramm segítségével határozzák meg a hőcseréket. Hőmérséklet intervallumokat határoznak meg, melyeken a hőcsere megtörténhet, a hőmérséklet intervallum maradék hője az alatta levő intervallumokba továbbítható [9, 15, 67]. Az algoritmikus módszerek fejlődésével és a számítási teljesítmény növekedésével egyre pontosabb, összetettebb feladatok megoldására nyílik lehetőség.

A pinch technológián és a szállítási modellen kívül számos más modell létezik HENS feladatok megoldására. Lin és Miller tabu-keresés módszerével oldották meg a hőcserélő hálózat tervezési feladatot [6]. A tabu-keresés a sztochasztikus optimalizálási módszerek családjába tartozik, amely az úgynevezett tabu-listákat használ rövid és hosszú távú memóriaként. Legegyszerűbb esetben egy megoldásból a szomszédos megoldások közül a legkedvezőbbe lép át, hacsak az nem szerepel a tabu listában. A listát minden lépés után aktualizálja, általában felveszi az előző megoldást, illetve törli a legrégebbit. A tabu-keresés jól alkalmazható HENS feladatok megoldására, mert nagy valószínűséggel megtalálja a globális optimumot, ugyanakkor kis számítási igényre van szüksége. Ravagnani és szerzőtársai genetikus algoritmus segítségével határozták meg a HENS feladat megoldását [78]. A megoldáshoz felhasználták a pinch analízist is. Módszerükben a minimális hőmérséklet különbség, a legtöbb eljárással

ellentétben, optimalizálási változó.

Szakaszos termelési rendszerekben a rendszer energia fogyasztása a teljes költség 5-10%-át is elérheti. Vaklieva-Bancheva és társai szakaszos, többcélú termelő rendszer költség optimális ütemezésére fejlesztettek ki egy MILP modellt [98]. A modellben a meghatározzák a berendezések között telepítendő hőcserélő egységeket és azok méretét.

Majozi szakaszos termelő rendszer hőintegrációjára egy „folytonos" MILP modellt dolgozott ki [48]. A modellt az SSN (*State Sequence Network*) ábrázolás alapján formalizálta. A modellben a maximális profitot adó ütemezést keresi, figyelembe véve a hőintegrációs részfeladat költségeit is.

Lee és Reklaitis periodikus, kampány módon működő szakaszos rendszer optimális hőintegrációt is figyelembe vevő ütemezését adták meg munkájukban [42, 43]. Egy MILP modellel határozzák meg a minimális energiát használó ütemezést. Hőcserét a rendszer anyagáramai között hozhatnak létre.

A hőintegrációval a szakirodalomban elsősorban folytonos folyamatok kapcsán találkozhatunk, mert a hőintegrációval foglalkozó módszerek feltételezik a rendszer állandósult állapotát, ami szakaszos rendszerekben nem teljesül. Ugyanakkor a szakaszos termelő struktúrák fontossága és a folyamatosan szigorodó környezetvédelmi szempontok megkövetelik, hogy a szakaszos termelési rendszerek emisszióját is csökkentsük hőintegrációval. Ehhez a hőintegrációs módszerek szakaszos folyamatokra való kiterjesztésére van szükség.

Számos különböző módszert találhatunk hőintegrációs, vagy szakaszos ütemezési feladatok megoldására. Kemp módszerében a direkt hőcsere helyett a hő tárolásával csökkenti a folyamatok összes hő fogyasztását [40]. Corominas és társai munkájában a több termékes szakaszos ütemezési feladat hő integrációját heurisztikus algoritmusokkal oldja meg, mely algoritmusokkal a műveletek kezdési idejét úgy határozza meg, hogy energia megtakarításra nyíljon lehetőség [16]. Ivanov kutatásaiban szakaszos reaktorok és hőcserélő hálózatok szintézisét integrálja [38].

6.1. táblázat. Szemléltető példa receptje.

Taszk	A termék		B termék	
	Berendezés	Idő (h)	Berendezés	Idő (h)
1	$E2$	8	$E3$	8
2	$E1$	4	$E1$	4
			$E4$	11

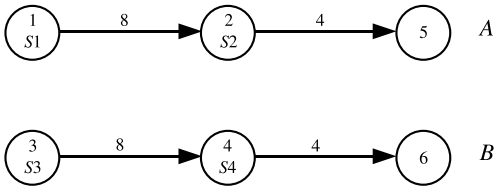
6.2. Ütemezési és hőintegrációs feladat kapcsolódása

Egy szemléltető példa segítségével mutatom be szakaszos folyamatok hőintegrációjának a szükségességét. Először a szemléltető példa minimális végrehajtási idejű ütemezését határozom meg, majd a kapott ütemezésre megadom a legjobb hőintegrációs lehetőséget. Bemutatom, hogy az ütemezés végrehajtási idejének növelése nem csökkenti a rendszer energia felhasználását, az ütemezésben lévő korlátok miatt nincsen lehetőség újabb hőcseréket létesíteni. Más ütemezéseket, struktúrákat használva – esetleg nagyobb végrehajtási idejűeket – ugyanakkor magasabb szintű hőintegrációt érhetünk el.

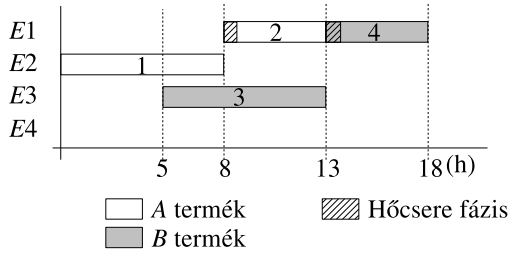
6.1 feladat

A 6.1 táblázat tartalmazza a gyártandó A és B termékek receptjeit. Mindkét terméket két egymás utáni taszkkal állíthatjuk elő. A táblázat tartalmazza a taszkok és a berendezések egymáshoz rendelésének szabályait. Például a B termék második taszkját az $E1$ berendezéssel 4 óra alatt, vagy $E4$ berendezéssel 11 óra alatt lehet végrehajtani. A 6.1. ábra a szemléltető példa recept-gráfját mutatja, ahol az $S1 = \{E2\}$, $S2 = \{E1\}$, $S3 = \{E3\}$ és $S4 = \{E1, E4\}$.

Tegyük fel, hogy az A termék második taszkjának bemenő anyagáramát hűteni, a B termék második taszkjának bemenő anyagáramát fűteni kell (az S-gráf 2-es és 4-es taszkjai) és ezek között a hőáramok között termodinamikai főtételei alapján hőcserét lehet végrehajtani. Az ütemezés alapján akkor tudunk a két hőáram között hőcserét végrehajtani, ha azok ugyanabban az időpontban jelen vannak a rendszerben. A két



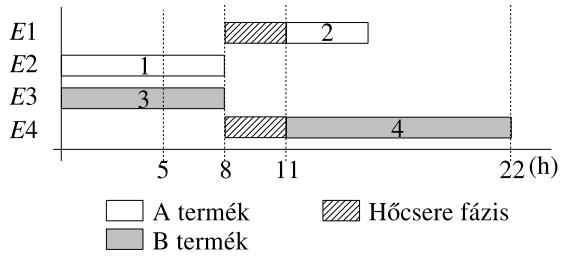
6.1. ábra. A 6.1. táblázat recept-gráfja.



6.2. ábra. Minimális végrehajtási idejű ütemezés Gantt-diagrammja.

hőáram közötti hőcserét 3 óra alatt lehet végrehajtani egy hőcserélő berendezéssel. Ha nem lehet a hőáramok közötti hőcserével kielégíteni a hűtés és fűtési igényt, akkor lehetőség van ezt külső energia szolgáltatót használva megtenni. Külső energia szolgáltatót használva 1 órára van szükség a bemenő anyagáramok hűtésére és fűtésére. A 6.2 ábra a minimális végrehajtási idejű ütemezés Gantt-diagrammját szemlélteti a külső energia szolgáltatókat használva kielégített hőcserék időigényeivel (hőcsere fázis). Ez a minimális végrehajtási idejű ütemezés nem teszi lehetővé, hogy a két hőáram közötti hőcserével módosítsuk az anyagáramok hőmérsékletét, mivel a hozzájuk tartozó taszkok megelőzik egymást. Hiába növeljük az ütemezés végrehajtási idejének korlátját, ennél az ütemezésnél hőcserét nem lehet létrehozni a taszkok bemenő anyagáramai között.

Az 6.3 ábra egy olyan, végrehajtási idő szempontjából nem optimális ütemezés Gantt-diagrammját tartalmazza, melynek a végrehajtási ideje nem minimális, azonban lehetővé teszi a taszkok bemenő anyagáramai közötti hőcserét. Az ábrán a vonalazott téglalappal jelölt hőcsere fázisok egyidejűsége miatt lehetőség van a külső



6.3. ábra. Egy megvalósítható ütemezés Gantt-diagrammja.

energia szolgáltatók helyett az olcsóbb, hőáramok közötti hőcsere végrehajtására.

6.3. Szakaszos ütemezési feladat hőintegrációja

Az ütemezési és hőintegrációs feladatban az optimális (minimális végrehajtási idejű) ütemezés mellett figyelembe kell venni a hőáramok közötti optimális hőcserét. A minimális végrehajtási idejű ütemezés általában korlátozza a hőcsere lehetőségét, ezért érdemes a végrehajtási idő minimalizálása helyett nagyobbnak engedni a végrehajtási időt és megpróbálni minél nagyobb hőintegrációt elérni az ütemezésben. Szakaszos folyamatok hőintegrációjaként egy olyan berendezés és hőcserélő berendezés ütemezést értünk, mely minimális külső energiát használ ugyanakkor az ütemezés végrehajtási ideje adott határidőn belüli (például 1,05-szöröse a minimális ütemezés végrehajtási idejének).

A szakaszos ütemezési feladatok hőintegrációjához a termékek gyártását ismertető recept mellett adottak a taszkok bemenő anyagáramainak a termodinamikai adatai. Ha egy taszk bemenő anyagáramának hőmérséklete nem megfelelő, akkor egy külső energia szolgáltatót, vagy másik hőáramot használva megfelelő hőmérsékletre kell hozni.

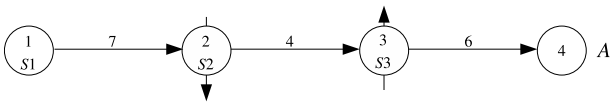
A taszkok működése ütemezés szempontjából felbontható két szakaszra, a hőcserélő fázisra és az aktív fázisra. A hőcserélő fázis az az időintervallum, amikor a taszk bejövő anyagáramait hőcsere (vagy hőcserék) segítségével az előírt hőmérsékletre hozzuk. A hőcserélő fázis után következik az aktív fázis, amikor a taszkhoz

rendelt berendezés végrehajtja a gyártási feladatot. Feltételezzük, hogy a berendezések fel vannak szerelve hőcserélő egységgel, hogy külső hőforrásból tudja fedezni a bemenő anyagáramok hűtési, fűtési igényeit, ha szükséges. Korlátlan mennyiségben áll rendelkezésünkre a külső hideg és meleg hőforrás. A hőcseréket adott felület nagyságú és a hőátadó tényezőjű hőcserélő berendezéssel lehet végrehajtani, melyekből meghatározható a hőcseréhez szükséges idő mennyiség. Feltételezzük, hogy hőcserélő berendezésben a hőcsere során átvitt hőmennyiség a hőcsere idejével arányos.

A hőcserélő berendezés felületének nagyságából, hőátadási tényezőjéből és a hőáramok adatai alapján bármely hőáram párra, melyek között a termodinamika törvényei alapján hőcsere jöhet létre, kiszámítható a hőcseréhez szükséges idő függetlenül az ütemezéstől. Adottak a rendelkezésre álló hideg- és melegenergia szolgáltatók, melyeket akkor használhatunk hőcserére, ha a belső anyagáramokkal a hőcsere nem valósítható meg.

6.4. Kapcsolódó komponens területek

A szakaszos ütemezési feladatok hőintegrációja feladat két komponens feladattípushoz kapcsolódik: a szakaszos folyamatok ütemezéséhez és a hőintegrációs feladathoz. Mindkét feladattípus tekinthető mester feladatnak, mely az optimum keresése során irányítja a másik komponens megoldás megtalálását. Mivel a vizsgált integrált feladatban az ütemezési feladat nagy komplexitásából ered az egész feladat nehézsége, ezért egy hatékony algoritmusban az ütemezésnek célszerű irányítania az integrált feladat megoldását. Egy szétválasztás és korlátozás algoritmust fejlesztettem ki az összetett feladat megoldására, melyben a mester feladat szerepét az ütemezés kapja. Minden részfeladatban a hőintegrációs feladat relaxációját vizsgálom az algoritmus korlátozási lépésében.



6.4. ábra. Recept-gráf kiegészítve hőfolyamatokkal.

6.4.1. Mester feladat: ütemezés

Szakaszos folyamatok ütemezési feladatát a recepttel adják meg, mely tartalmazza a termékekből ütemezendő batchek számát és a taszkokhoz hozzárendelhető berendezések halmazait. Ütemezésnél általában a cél, hogy megadjunk egy olyan ütemezést, melynek végrehajtási ideje minimális.

Az S-gráf algoritmus [84, 86] egy hatékony feladat és megoldás ábrázolás és algoritmus NIS tárolási politika esetén. Az S-gráf algoritmus részletes bemutatása a 3. fejezetben található. Mester feladat megoldására az EQ-SG algoritmust használom. Az EQ-SG algoritmus *EQ-korlátozás* eljárását építem át az ütemezési feladat és a relaxált hőintegrációs feladat kezelésére.

6.4.2. Szakaszos folyamatok hőintegrációja

Szakaszos folyamatokhoz kapcsolódó hőintegrációs részfeladatot a következőképpen definiálhatjuk. A hőintegrációs részfeladathoz adva van a taszkok bemenő anyagáramainak termodinamikai adatai, tehát a hideg- és melegáramok, melyek a receptben jelen vannak. A bemenő anyagáramokhoz adva vannak a hőmérséklet adatok és az anyagáramok áramlásisebességei (*flowrate*). Továbbá adva vannak a rendelkezésre álló hőcserélő berendezések, a hőcserélő berendezések felülete és hőátadási tényezője.

Ha egy taszk bemenő anyagáramának hőmérséklete nem megfelelő a receptben, és hőcsere útján más hőmérsékletre kell hozni, akkor a hőcsereigény helyét a recept-gráfban ábrázoljuk. A 6.4 ábra az A termék előállítását ábrázoló recept-gráfot tartalmazza. A recept-gráfban az anyagáramok hűtésének és fűtésének a szükségességét a taszkoknál függőleges nyilakkal jelöljük. A felfelé mutató nyíl a fűtést (3-as taszk) a lefele mutató nyíl a hűtést jelzi (2-es taszk).

6.5. Szakaszos folyamatok hőintegrációjára kidolgozott algoritmus

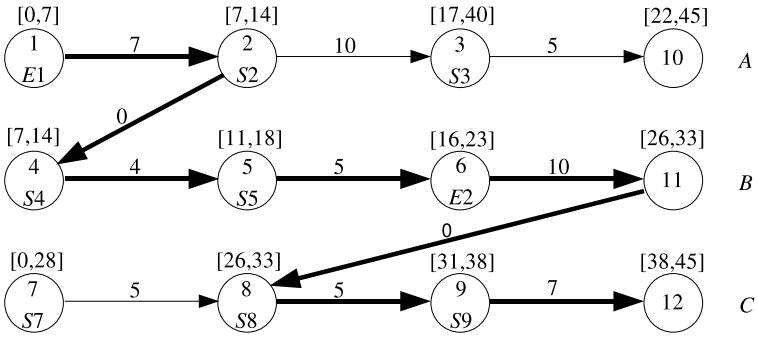
Az EQ-SG algoritmus a minimális végrehajtási idejű ütemezést határozza meg. A hőintegrációs feladatban egy olyan berendezés és hőcserélő berendezés ütemezést kell meghatározni, mely minimális költségű külső energiát használ és az ütemezés végrehajtási ideje nem nagyobb egy adott korlátnál. Az integrált feladat megoldására kidolgozott módszer szétválasztás és korlátozás elvén működik. A szétválasztás része felelős a berendezések ütemezéséért. A korlátozás része ellenőrzi a részütemezés megvalósíthatóságát és meghatározza a szükséges külső energia költségek alsó korlátját.

Az integrált feladat fő nehézsége, hogy a berendezések egy olyan ütemezését határozzuk meg, melyben lehetőség van minél több hőáramok közötti hőcserére. Ha egy hideg- és melegáram termodinamikai feltételek alapján összekapcsolható egy hőcserélő berendezésben, az ütemezés alapján csak akkor kapcsolhatóak össze, ha időben egyszerre vannak jelen. A lehetséges párhuzamos hőáramok kiszűréséhez bevezettem az időintervallumokat az S-gráfokba.

6.5.1. Időintervallumok időben párhuzamos taszkok kezelésére

Az S-gráf módszertant időintervallumokkal kiegészítve követhetjük a lehetséges párhuzamos hideg- és melegáramokat. Hőcsere akkor jöhet létre két taszk megfelelő hideg- és melegárama között, ha a két taszk anyagáramainak hőmérsékletei termodinamika törvényei alapján megfelelőek a hőcserére. Egy taszkhoz rendelt időintervallum tartalmazza az összes lehetséges időpontot, mikor egy taszk elkezdődhet. Formálisan, ha az i taszk a $G(N, A_1, A_2)$ S-gráf taszk csúcsa és a $[t_i, T_i]$ intervallum az i taszkhoz rendelt időintervallum, akkor létezik egy olyan $t \in [t_i, T_i]$ időpont, mikor az i taszk elkezdődhet és az ütemezés határideje teljesül.

Az S-gráf taszkjainak időintervallumai meghatározhatóak a leghosszabb út kereső algoritmussal. Ezzel az algoritmussal meghatározhatjuk minden taszkhoz azt a legkorábbi időpontot, amikor elkezdődhetnek az ütemezés kezdéséhez képest. Az algoritmus az éleken visszafelé haladva meghatározza a taszkoknak azt a legkésőbbi



6.5. ábra. S-gráf időintervallumokkal.

kezdési idejét, mellyel még nem sérti meg az ütemezés végrehajtási idejének korlátját. A 6.5 ábra egy S-gráfot mutat időintervallumokkal kiegészítve. Az S-gráfban a vastag élek mutatják a leghosszabb utat. Ha a gyártás határideje 45 óra, akkor az időintervallumokat az S-gráfról leolvashatjuk.

6.5.2. Szétválasztás eljárás

A kifejlesztett szétválasztás és korlátozás algoritmus relaxált részfeladatok megoldásán keresztül egy olyan ütemezési-gráfot keres, melyben úgy lehet ütemezni a hőcseréket, hogy közben minimális külső energiát használ a rendszer. Ezeket a részfeladatokat egy keresési fában lehet rendszerezni. A recept-gráf a keresési fa gyökeréhez tartozik. Minden részfeladatnál kiválasztunk egy berendezést, és generáljuk a részfeladat gyermek részfeladatait figyelembe véve az összes lehetséges ütemezésre váró taszkot, melyet végre lehet hajtani a berendezéssel. A gyermek részfeladatokban ütemezzük a kiválasztott berendezéshez a megfelelő ütemezésre váró taszkot (a „berendezés szempontú" stratégiának megfelelően a berendezés utolsó taszkja lesz). Az S-gráf keretalgoritmusban és a jelenlegi feladatnál sincs meghatározva a keresési és kiválasztási stratégia a szétválasztás során, hanem az implementációra van bízva. Az ütemezés hatékonysága szempontjából érdemes a fontos, leterhelt berendezések ütemezésével kezdeni a keresést, melyek az ütemezés „jóságát" már az algoritmus futása elején meghatározhatják.

```
procedure SCH-HENS-korlátozás(PP)
begin
    kör_kereső(G(PP));
    if körmentes
      begin
        időintervallum_frissítés(G(PP));
        if megvalósítható
            bound :=hensmodell(G(PP));
        else
            bound := ∞;
      end
    else
        bound := ∞;
    return bound;
end.
```

6.6. ábra. Az *SCH-HENS-korlátozás* eljárás.

6.5.3. Korlátozás eljárás

A korlátozás lépésben a részfeladat megvalósíthatóságát vizsgáljuk és egy alsó korlátot számítunk a relaxált hőintegrációs feladat energia költségére. A részfeladat megvalósítható, ha a hozzá tartozó S-gráf körmentes és az S-gráf leghosszabb útja, azaz a belőle származó levél részfeladatok végrehajtási idejének alsó korlátja, kisebb mint a gyártás befejezésére adott határidő.

A hőcsere költségének alsó korlátját minden részfeladat számára meg kell határozni. Az idő intervallumokat használom a taszkok lehetséges kezdési idejének a becslésére. Az idő intervallumokkal csökkenthető az olyan lehetségesen párhuzamos hőáramok száma, melyeket figyelembe kell venni a relaxált modellben. A 6.6 ábra az integrált feladathoz kidolgozott korlátozás eljárást tartalmazza.

A *kör_kereső* függvény a részfeladat S-gráfjának körmentességét ellenőrzi (B. függelék). Ha az S-gráf körmentes, akkor a *körmentes* változó értéke igaz. Az *intervallum_frissítés* függvény a leghosszabb út kereső algoritmus kétszeri alkalmazásával

meghatározza a taszkok időintervallumait. A függvény az intervallumok meghatározása során ellenőrzi az ütemezés leghosszabb útjának értékét. Amennyiben a végrehajtási időre megadott korlátnál kisebb, akkor a *megvalósítható* változó értékét igazra állítva tér vissza a függvény. A *hensmodell* a relaxált hőintegrációs feladat alsó korlátját határozza meg.

A HENS relaxált modellben a H és C halmazok tartalmazzák a recept meleg- és hidegáramait. A $h \in H$ melegáramra h_{in} és h_{out} jelölje a melegáram kezdeti és végső hőmérsékletét ($h_{out}<h_{in}$). Hasonlóan, ha $c \in C$ hidegáram, akkor jelölje c_{in} és c_{out} a hidegáram kezdeti és végső hőmérsékletét ($c_{in}<c_{out}$).

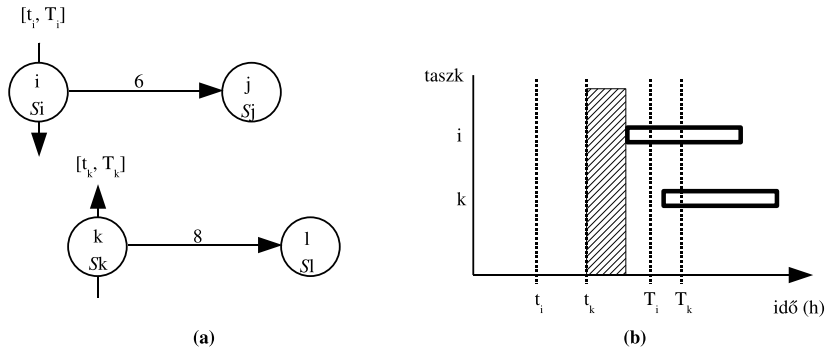
Tartalmazza a HP halmaz az összes olyan meleg- és hidegáram párt, melyeket az adott részfeladatban figyelembe kell venni hőcsere során. Tehát ha a $(h, c) \in HP$ pár, akkor a h és c hőáramok között a termodinamika második törvénye alapján hőcsere jöhet létre ($h \in H, c \in C$), és a részfeladat S-gráfja alapján a meleg- és hidegáram párhuzamosak lehetnek (megfelelő időintervallumok nem diszjunktak).

A 6.7(a) ábra egy S-gráfot négy taszkkal tartalmaz. Az i és k taszk hőárama párhuzamos, ha a $[t_i, T_i]$ és $[t_k, T_k]$ halmazok nem diszjunktak. Ez egy szükséges feltétel a párhuzamosságra. A 6.7(b) ábra az idő intervallumok egy elrendezését ábrázolják. A vonalazott terület a lehetséges hőcsere fázist jelöli. A két hőáram között a legkorábbi olyan időpontban kezdődhet el, amikor már mindkét taszk végrehajtható azaz a $max(t_i, t_k)$ időpontban.

A relaxált modell változói

Jelölje y_{hc} bináris változó a $(h, c) \in HP$ párral jelölt meleg- és hidegáram pár közötti hőcserét az optimális hőcserélő hálózatban. Ha az $y_{hc} = 1$, akkor a hálózat tartalmazza, ha $y_{hc} = 0$ akkor nem tartalmazza. Jelölje a nemnegatív t_{hc} folytonos változó a h és c hőáramok közötti hőcsere idejét ($\forall (h, c) \in HP$). Jelölje az x_{hc} folytonos változó azt a potenciális időpontot, amikor a h melegáramtól a c hidegáram irányába a hőcsere elkezdődhet, x_{ch} pedig a c hidegáram irányából h melegáram irányába induló hőcsere lehetséges időpontja.

Minden hőáramra vezessünk be egy folytonos változót meleg- és hidegenergia szolgáltató irányában létrejött hőcserék hosszának jelölésére. Ha $c \in C$ hidegáram, akkor



6.7. ábra. Párhuzamos meleg- és hidegáramok (a vonalazott terület a hőcsere lehetséges idejét jelöli).

jelölje $t_{c,util}$ a külső forrás és a c hidegáram közötti hőcsere időhosszát. Hasonlóan jelölje $t_{h,util}$ a $h \in H$ melegáram és a szolgáltató közötti hőcsere időhosszát. Jelöljék a nemnegatív U_h és U_c változók a szolgáltatók és a h és c hőáramok közötti hőmennyiség cserét ($k \in H$, $c \in C$).

Jelölje a $G(N, A_1, A_2)$ a részfeladathoz tartozó S-gráfot. Minden $i \in N$ csúcsra a nemnegatív x_i változó a taszk-csúcshoz tartozó taszk lehetséges kezdési idejét, vagy a termék-csúcshoz tartozó termék lehetséges elkészülési idejét.

A modell korlátozásai

Mivel egy hőáram legfeljebb egy másik hőárammal való hőcserében szerepelhet, ezért teljesülnie kell a

$$\sum_c y_{hc} \leq 1, \text{ minden } h \in H \tag{6.5.1}$$

$$\sum_h y_{hc} \leq 1, \text{ minden } c \in C. \tag{6.5.2}$$

Ha egy hőcsere nincs benne az optimális hálózatban, akkor a hőcsere időhossza nulla, azaz

$$t_{hc} \leq M y_{hc}, \text{ minden } (h, c) \in HP \tag{6.5.3}$$

ahol az M egy kellően nagy konstans.

Minden $h \in H$-ra jelölje a QF_h konstans a h melegáram hőtartalom leadását. Hasonlóan minden $c \in C$-re jelölje a QF_c konstans a c hidegáram hőtartalom felvételét. A QF_h és QF_c-t a fajhő, az anyagáram folyamértéke és a hőáram kezdő és végső hőmérséklet különbségének a szorzata adja. A hőáram hőtartalom leadása vagy felvétele a többi hőáram és az energia szolgáltató irányában létrejött hőcserék összegével számolható, azaz

$$\sum_c Q_{hc} + U_h = QF_h, \text{ minden } h \in H \tag{6.5.4}$$

$$\sum_h Q_{hc} + U_c = QF_c, \text{ minden } c \in H \tag{6.5.5}$$

Feltételeztük, hogy a hőáram hőtartalom leadása, vagy felvétele arányos a hőcsere időhosszával, azaz

$$Q_{hc} = D_{hc} t_{hc}, \text{ minden } (h,c) \in HP \tag{6.5.6}$$

ahol a D_{hc} konstans a hőcsere sebessége. A $D_{hc} = U_{hc} LMTD_{hc} AREA$, ahol U_{hc} a hőcsere hőátadási tényezője, $LMTD_{hc}$ a logaritmikus hőmérséklet különbség átlag a h meleg és c hidegáramra, az $AREA$ pedig a hőcserélő berendezések felülete.

A (6.5.6) képlethez hasonlóan a külső energia szolgáltató és a hőáram közötti hőcsere során az elvonandó, illetve betáplálandó hő mennyisége arányos a hőáram és az energia szolgáltató közti hőcsere időhosszával ($t_{c,util}$ és $t_{c,util}$), azaz

$$U_c = D_c t_{c,util}, \text{ minden } c \in C \tag{6.5.7}$$

$$U_h = D_h t_{h,util}, \text{ minden } h \in H \tag{6.5.8}$$

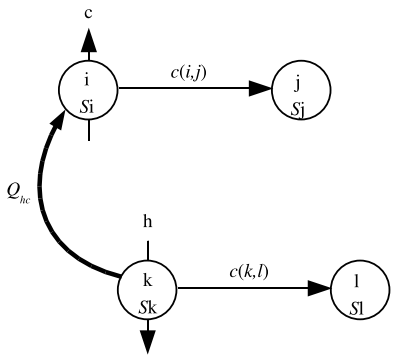
ahol D_c és D_h paraméterek a szolgáltató és a hőáram közötti hőmérséklet különbség függvénye.

Az ütemezés végrehajtási idejének korlátja miatt az x_i változók nem lehetnek nagyobbak mint a korlát (MS), azaz

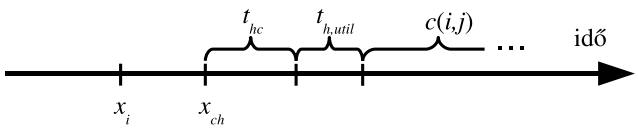
$$0 \leq x_i \leq MS, \text{ minden } i \in N \tag{6.5.9}$$

Ha az $i, j \in N$ a részfeladat S-gráfjának csúcsai, melyek között $c(i,j)$ súlyú él van és az i taszkhoz nem tartozik hőáram, akkor a j taszk nem kezdődhet korábban, mint az i taszk kezdési idejének és végrehajtási idejének összege, azaz

$$x_i + c(i,j) \leq x_j, \text{ minden } (i,j) \in A_1 \cup A_2 \tag{6.5.10}$$



6.8. ábra. S-gráf lehetséges hőcserével.



6.9. ábra. Az i taszk tevékenységeinek sorozata hőcsere esetén.

A 6.8 ábra egy S-gráfot tartalmaz egy lehetséges hőcserével a $(h, c) \in HP$ pár között. A 6.9 ábra az i taszk tevékenységeinek sorozatát mutatja. A taszk bemenő anyaga az x_i időpont áll rendelkezésre az i taszkhoz rendelt berendezésben. A hőcsere az x_{ch} időpontban kezdődhet el a h melegáram és a c hidegáram között, a hőcsere t_{ch} ideig tart. A h hőárammal való hőcsere után, ha szükséges a meleg energia szolgáltatót használva $t_{c,util}$ idő alatt éri el a kívánt hőmérsékletet a c hidegáram. A tevékenységsorrend a c és h hőáramra a (6.5.11)–(6.5.14) képletekkel írható le.

$$x_i \leq x_{ch} \tag{6.5.11}$$

$$x_{ch} + t_{hc} + t_{c,util} + c(i,j) \leq x_j \tag{6.5.12}$$

$$x_k \leq x_{hc} \tag{6.5.13}$$

$$x_{hc} + t_{hc} + t_{h,util} + c(k,l) \leq x_l \tag{6.5.14}$$

Ha egy h melegáram és c hidegáram közötti hőcsere a hőcserélő hálózat része, akkor a két áram közötti hőcsere egyszerre kell, hogy elkezdőjön, azaz teljesülnie kell az $x_{hc} = x_{ch}$ egyenlőségnek. A (6.5.15) képlet a h és c párhuzamosságát biztosítja, azaz

$$-M(1 - y_{hc}) \leq x_{hc} - x_{ch} \leq M(1 - y_{hc}), \text{ minden } (h, c) \in HP \tag{6.5.15}$$

A relaxált modell célfüggvénye

A relaxált modellben a célfüggvény a külső energia szolgáltató használatának minimalizálása. A célfüggvény a 6.5.16 képlettel írható le.

$$\min(K_c \sum_c U_c + K_h \sum_h U_h) \tag{6.5.16}$$

ahol a K_c és K_h a hideg és meleg energia költségei.

6.6. Példa szakaszos ütemezési feladat hőintegrációjára

Szakaszos ütemezési feladat hőintegrációval való együttes megoldásával szemléltetem az algoritmus működését. A feladatban négy batchnyi A és három batchnyi B és két batchnyi C terméket kell előállítani. A feladat receptje a 6.2 táblázatban található. A recepthez tartozó hideg- és melegáramokat, a 6.3 táblázat tartalmazza. A 6.10 ábra a feladat recept-gráfját ábrázolja bejelölve a receptben szereplő hőáramok helyét.

$5m^2$ felületű, $1500W/m^2K$ hőátadási tényezőjű hőcserélő berendezéseket használhatunk a hőáramok közötti hőcserére. Külső hideg és meleg szolgáltatótól $283K$ és $473K$ hőmérsékletű hőforrás áll rendelkezésünkre mennyiségi korlátozás nélkül. A hideg és meleg energia szolgáltató költsége 1, azaz $K_c = 1$ és $K_h = 1$.

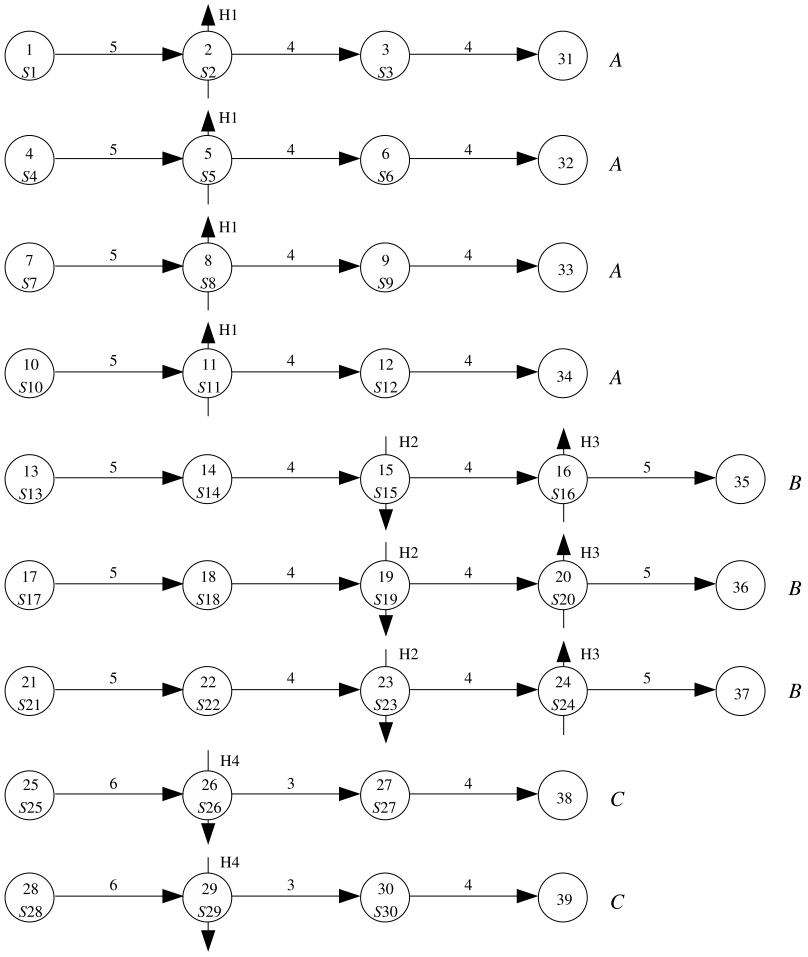
A minimális végrehajtási idejű ütemezés $31,1$ óra hosszú. Ehhez az ütemezéshez 3100 MJ energiát kell a külső forrásból pótolni. Ha a végrehajtási idő korlátját

6.2. táblázat. A hőintegrációs feladat receptje.

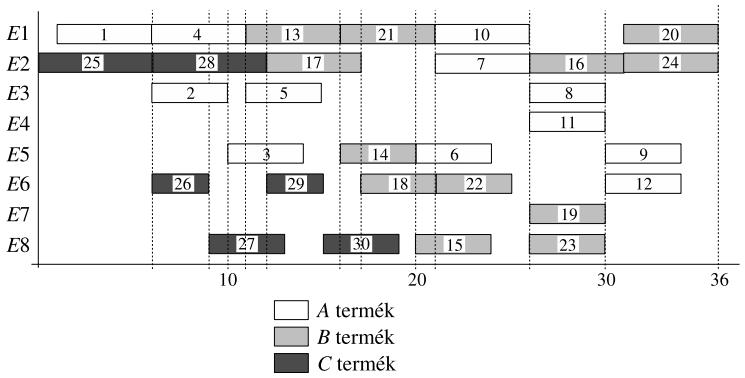
Taszk	A termék		B termék		C termék	
	Berendezés	Idő (h)	Berendezés	Idő (h)	Berendezés	Idő (h)
1	$E1$	5	$E1$	5	$E1$	6
	$E2$	5	$E2$	5	$E2$	6
2	$E3$	4	$E5$	4	$E5$	3
	$E4$	4	$E6$	4	$E6$	3
3	$E5$	4	$E7$	4	$E7$	4
	$E6$	4	$E8$	4	$E8$	4
4			$E1$	5		
			$E2$	5		

6.3. táblázat. A hőintegrációs feladat hőáramai.

Hőáram azonosító	Kezdő hőmérséklet (K)	Végső hőmérséklet (K)	Hőtartalom (MJ)	Taszk-csúcs azonosító
H1	313	393	400	2,5,8,11
H2	413	323	200	15,19,23
H3	353	403	100	16,20, 24
H4	423	313	300	26,29



6.10. ábra. A 6.1. feladat recept-gráfja.



6.11. ábra. A feladat optimális ütemezésének Gantt-diagrammja.

6.4. táblázat. A feladat optimális megoldásában levő hőcserék.

Azonosító	Hőcsere (taszk-csúcsok)	Kezdés (h)	Időtartam (h)
1	$19 \rightarrow 8$	25,933	0,428
2	$23 \rightarrow 11$	26	0,428
3	$26 \rightarrow 2$	9	0,375
4	$29 \rightarrow 5$	12	0,375

36 órára növeljük, akkor az optimális külső energia használat lecsökken 1100 MJ-ra. A 6.11 ábra az optimális (minimális) külső energiát használó ütemezés Gantt-diagrammját mutatja. A Gantt-diagrammhoz tartozó hőcserék időzítését a 6.4 táblázat tartalmazza.

6.7. Összefoglalás

Szakaszos gyártási folyamat egyidejű ütemezésére és hőintegrációjára kifejlesztettem egy algoritmust. Az integrált feladat megoldására kidolgozott algoritmus az S-gráf módszertan [86] minimális végrehajtási idejű ütemezése helyett a minimális energia költségű ütemezés meghatározását végzi, miközben a gyártási folyamat végrehajtási ideje elfogadható marad.

A fejezet témájához kapcsolódó publikációim

Nemzetközi folyóirat cikk

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

Nemzetközi konferencia előadás

Adonyi, R., J. Romero, L. Puigjaner, and F. Friedler, Heat Integration for Batch Processes presented at PRES'03, Hamilton, Ontario, Canada, October 26-29, 2003.

Adonyi, R., J. Romero, L. Puigjaner, and F. Friedler, Incorporating Heat Integration in Batch Process Scheduling, presented at the PRES'02 (5th Conference on Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction), Praha, Czech Republic, August 25-29, 2002.

7. fejezet

Új tudományos eredmények

Az értekezés új tudományos eredményeinek tézisszerű összefoglalása:

1. Új szétválasztási eljárás az S-gráf módszertanhoz

   1.1. Új szétválasztási algoritmust dolgoztam ki az S-gráf módszertan alapalgoritmusához, amely a korábban alkalmazott „berendezés szempontú döntések" helyett a „taszk szempontú" döntéseket használja.

   1.2. Beépítettem a „taszk szempontú" döntéseket használó szétválasztási eljárást az S-gráf módszertanba. Bemutattam az új ütemező algoritmus működését és ismertetem mely esetekben előnyös az alkalmazása más módszerekkel szemben.

2. Optimális ütemezés berendezések tisztítását is figyelembe véve

   Az S-gráf módszertant kiterjesztettem a berendezések ütemezésétől függő tisztítási költségek kezelésére. Bevezettem egy új korlátozás eljárást, mely minimális végrehajtási idő mellett figyelembe veszi az ütemezéshez kapcsolódó tisztítási költséget. Az algoritmust ipari feladat megoldásával szemléltettem.

3. Optimális ütemezés és hőintegráció

   3.1. Új matematikai modellt dolgoztam ki ütemezés és energia használat együttes optimalizálására.

3.2. Kidolgoztam az algoritmust az ütemezési és hőintegrációs feladatra, feladat megoldásával illusztráltam működését.

8. fejezet

Major results and summary of accomplishments

Brief summary of the new scientific results derived from the thesis:

1. New branching procedure for the S-graph framework

   1.1. A new branching procedure has been developed for the S-graph framework. The new procedure uses the "task aspect decisions" instead of the formerly developed "equipment aspect decisions".

   1.2. The branching procedure using the "task aspect decisions" has been integrated into the S-graph framework. The advantages of the new procedure have been demonstrated through several case studies, providing a comparison with the results obtained by the previous procedure.

2. Optimal scheduling accounting for cleaning costs

   The S-graph framework has been extended to consider the equipment changeovers with cleaning costs. A new bounding procedure has been introduced, that considers the minimal makespan and the cleaning costs connected to the scheduling. The algorithm has been demonstrated by solving an industrial example.

3. Optimal scheduling and heat integration

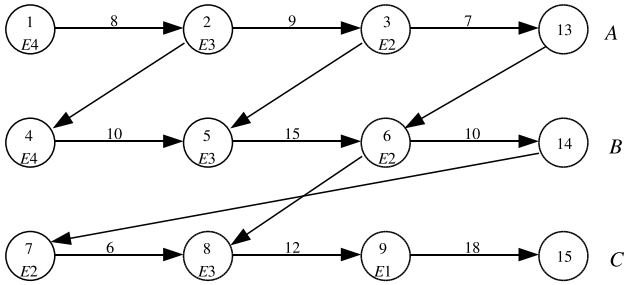
3.1. A new mathematical model has been developed to solve combined scheduling and the heat integration problems.

3.2. An algorithm has been developed to solve the combined scheduling and heat integration problems. The work of the algorithm has been demonstrated through the solution of examples.

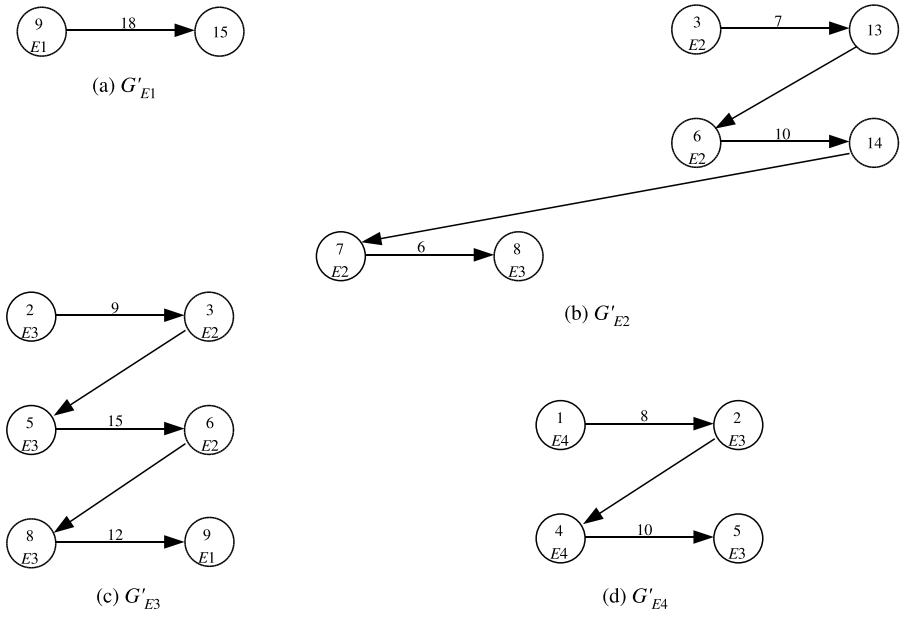
A. Függelék

Komponens-gráf

Az A.1 ábrán látható ütemezési-gráf $E1$, $E2$, $E3$ és $E4$ berendezéseihez tartozó komponens-gráfokat az A.2 ábra tartalmazza.



A.1. ábra. Ütemezési-gráf a komponens-gráfok bemutatásához.



(a) G'_{E1} (b) G'_{E2} (c) G'_{E3} (d) G'_{E4}

A.2. ábra. A G'_{E1}, G'_{E2}, G'_{E3} és G'_{E4} komponens-gráfok.

B. Függelék

Körkereső algoritmus

A korlátozási lépés hatékonysága érdekében a körkereső algoritmus megvalósításakor érdemes figyelembe venni, hogy minden egyes részfeladat keletkezése során legfeljebb egy új ütemezési-él hozzáadása történik meg egy körmentes S-gráfhoz. Ez azt jelenti, hogy elég olyan köröket keresni a részfeladatban, melyek az új ütemezési-élt tartalmazzák, tehát elég az utoljára beütemezett taszk-csúcsból kiindulva „lokálisan" kört keresni. A B.1 ábrán az i taszk-csúcsból induló az S-gráfot bejáró körkeresőre láthatunk egy megvalósítást.

```
procedure kör_kereső(i,LIST)
jelölés:
      pred(i): azon csúcsok halmaza, melyekből van él i csúcsba
begin
      if pred(i) = ∅ then
            return no_cycle;
      for all j ∈ pred(i)
         begin
            if j ∈ LIST then
                  return cycle;
            LIST := LIST ∪ {j};
            kör_kereső(j,LIST);
            if cycle then
                  return cycle;
            LIST := LIST \ {j};
         end
      return no_cycle;
end.
```

B.1. ábra. A körkereső algoritmus.

C. Függelék

Leghosszabb-út kereső algoritmus

Az értekezésben a szétválasztás és korlátozás algoritmus a leghosszabb út kereső algoritmust használhatja a részfeladat végrehajtási idejének alsó korlátának meghatározására. A C.1. ábra a leghosszabb út kereső algoritmust tartalmazza. Az algoritmus a paraméterként kapott $G(N, A)$ irányított, körmentes gráf leghosszabb útjainak értékét határozza meg a gráf csúcsaiban ($d(i)$ értéke, ahol $i \in N$).

```
procedure leghosszabb_út_kereső(G(N, A))
jelölés:
    pred(i): azon csúcsok halmaza, melyekből van él i csúcsba
begin
    for all i ∈ N
        d(i) := 0;
    for all i ∈ N
      begin
        LIST := {i};
        longest := 0;
        while LIST ≠ ∅
          begin
            LIST := LIST \ {i};
            for all (i, j) ∈ A
              begin
                if d(j) < d(i) + c(i, j) then
                  begin
                    d(j) = d(i) + c(i, j);
                    longest := max(longest, d(j));
                    if j ∉ LIST then
                        LIST := LIST ∪ {j};
                  end
              end
          end
      end
    return no_cycle;
end.
```

C.1. ábra. A leghosszabb út kereső algoritmus.

D. Függelék

Lineáris programozási modell a részfeladat alsó korlátjának számítására

A modell felírásához tekintsük a vizsgált részfeladat körmentes S-gráfját. Jelölje L_j a j csúcsba irányított éleken át vezető leghosszabb út értékét az aktuális részproblémához tartozó S-gráfban. Jelölje c_i az i berendezés utolsónak ütemezett taszkjának befejezési idejét. A c_i értéke a D.0.1 képlettel számítható.

$$c_i = max(max_{j \in M_i}(L_j + t_{ij}), min_{j \in \overline{M_i}}(L_j)) + \sum_{j \in \overline{M_i}} t_{ij} \qquad \text{(D.0.1)}$$

ahol az M_i halmaz az i berendezéshez rendelt beütemezett taszkokat tartalmazza, az $\overline{M_i}$ halmaz pedig azokat a nem ütemezett taszkokat, melyeket egyedül az i berendezés tud végrehajtani. A t_{ij} jelöli a j taszk végrehajtási idejét, ha az i berendezéssel hajtjuk végre.

Tartalmazza az $\overline{N}$ halmaz azokat a nem ütemezett csúcsokat az S-gráfban, amelyekhez több mint egy berendezés áll rendelkezésre. Jelölje az x_{ij} nemnegatív folytonos változó az i berendezés j taszkhoz való hozzárendelésének hosszát, ahol $i = 1, 2, \cdots, n$ és $j \in \overline{N}$. Így az i berendezés legkorábbi befejezési ideje $c_i + \sum_{j \in \overline{N}} x_{ij}$ képlettel számítható ($i = 1, 2, \cdots, n$), mely alsó korlátja a részprobléma végrehajtási idejének, amit jelöljön az X változó. Minden $j \in \overline{N}$ taszkot végre kell hajtani

D.1. táblázat. A szemléltető példa receptje.

Taszk	A termék		B termék		C termék	
	Berendezés	Idő (h)	Berendezés	Idő (h)	Berendezés	Idő (h)
1	$E1$	8	$E1$	9	$E1$	7
			$E2$	11	$E2$	7
2	$E2$	15	$E3$	5	$E3$	4
	$E3$	5				

legalább egy berendezéssel, azaz $\sum_{i \in S_j} \frac{x_{ij}}{t_{ij}} \geq 1$.

A D.0.2-D.0.5 képletekkel felírt LP modell X megoldása alsó korlátja az aktuális részproblémának.

$$\mathbf{min} X \tag{D.0.2}$$

feltéve, hogy

$$c_i + \sum_{j \in \overline{N}} x_{ij} \leq X, \ i = 1, 2, \cdots, n \tag{D.0.3}$$

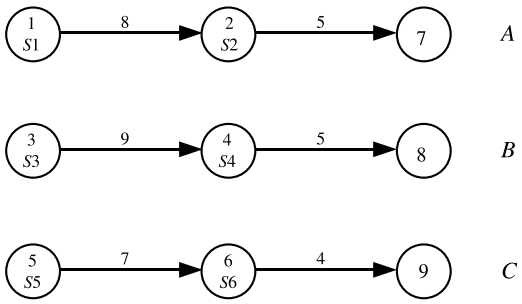
$$\sum_{i \in S_j} \frac{x_{ij}}{t_{ij}} \geq 1, \ j \in \overline{N} \tag{D.0.4}$$

$$x_{ij} \geq 0, \text{ ahol } i = 1, 2, \cdots, n, j \in \overline{N} \tag{D.0.5}$$

ahol n a berendezések száma, S_j halmaz a j taszkot végrehajtható berendezések halmaza.

Szemléltető példa az LP modellre

A következő szemléltető példán az LP modell felírását mutatom meg egy ütemezési részfeladatra. Tegyük fel, hogy az A, B és C termékeket kell előállítanunk az $E1$, $E2$ és $E3$ berendezésekkel. A termékek receptje a D.1. táblázatban található. A feladat recept-gráfja egy-egy batchnyi termékre a D.1 ábrán található. A recept-gráfban $S1 = \{E1\}$, $S2 = \{E2, E3\}$, $S3 = \{E1, E2\}$, $S4 = \{E3\}$, $S5 = \{E1, E2\}$ és $S6 = \{E3\}$.



D.1. ábra. A szemléltető példa recept-gráfja.

A szemléltető példa gyökér részfeladatának alsó korlátjának meghatározására megadom az LP modellt. Az LP modell megoldását összehasonlítom a leghosszabb út kereső algoritmus által kapott alsó korláttal. A gyökér részfeladathoz tartozó S-gráf megegyezik a szemléltető példa recept-gráfjával, amely gráf még nem tartalmaz berendezésekre vonatkozó döntéseket. Így az $M_i = \varnothing$ minden $i = 1, 2, 3$-ra. A D.1. táblázat alapján $\overline{M_1} = \{1\}$, $\overline{M_2} = \varnothing$ és $\overline{M_3} = \{4, 6\}$. A D.0.1 képlet alapján $c_1 = 8$, $c_2 = 0$ és $c_3 = 16$. A gyökér részfeladathoz tartozó LP modell a D.0.2-D.0.5 képletek alapján a D.0.6-D.0.13 modellt eredményezik.

$$\mathbf{min} X \tag{D.0.6}$$

feltéve, hogy

$$8 + x_{13} + x_{15} \leq X \tag{D.0.7}$$

$$x_{22} + x_{23} + x_{25} \leq X \tag{D.0.8}$$

$$16 + x_{32} \leq X \tag{D.0.9}$$

$$\frac{x_{13}}{9} + \frac{x_{23}}{11} \geq 1 \tag{D.0.10}$$

$$\frac{x_{15}}{7} + \frac{x_{25}}{7} \geq 1 \tag{D.0.11}$$

$$\frac{x_{22}}{15} + \frac{x_{32}}{5} \geq 1 \tag{D.0.12}$$

$$x_{13}, x_{15}, x_{22}, x_{23}, x_{25}, x_{32} \geq 0 \tag{D.0.13}$$

Az LP modellből számított alsó korlát értéke $X = 17,4$ óra. A leghosszabb út kereső algoritmusból számított alsó korlát értéke 14 óra.

Ahogy a szemléltető példa is mutatja az LP modellel élesebb alsó korlátot kaphatunk a részfeladatokra, mint a leghosszabb út kereső algoritmussal. Azonban az LP modell felépítése és megoldása sokkal nagyobb számítási teljesítményt igényel, mint a leghosszabb út kereső algoritmussal számított alsó korlát. A levél közeli részfeladatok esetén a két különböző módon számolt alsó korlát között a különbség egyre jobban csökken, ezért érdemes gyökér közeli részproblémák esetén az LP modellt, míg levél közeli részproblémák esetén a leghosszabb út kereső algoritmust használni az alsó korlát számítására.

Irodalomjegyzék

[1] A. Allahverdi and F. S. Al-Anzi, *A branch-and-bound algorithm for three-machine flowshop scheduling problem to minimize total completion time with separate setup times*, European Journal of Operational Research **169** (2006), 767–780.

[2] M. Dell' Amico and S. Martello, *Open shop, satellite communication and a theorem by Egerváry (1931)*, Operations Research Letters **18** (1996), 207–211.

[3] E. M. Arkin, M. A. Bender, J. S. B. Mitchell, and S. S. Skiena, *The lazy bureaucrat scheduling problem*, Information and Computation **184** (2003), 129–146.

[4] Y. Azar and O. Regev, *On-line bin-stretching*, Theoretical Computer Science **268** (2001), 17–41.

[5] A. Azaron, H. Katagiri, M.i Sakawa, K. Kato, and A. Memariani, *A multi-objective resource allocation problem in pert networks*, European Journal of Operational Research **172** (2006), 838–854.

[6] B. and D. C. Miller, *Solving heat exchanger network synthesis problems with tabu search*, Computers & Chemical Engineering **28** (2004), 1451–1464.

[7] A. Bachman, A. Janiak, and M. Y. Kovalyov, *Minimizing the total weighted completion time of deteriorating jobs*, Information Processing Letters **81** (2002), 81–84.

[8] J. Bank and F. Werner, *Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties*, Mathematical and Computer Modelling **33** (2001), 363–383.

[9] A. Barbaro and M. J. Bagajewicz, *New rigorous one-step MILP formulation for heat exchanger network synthesis*, Computers & Chemical Engineering **29** (2005), 1945–1976.

[10] W. G. Sr. Bistline, S. Banerjee, and A. Banerjee, *RTSS: An interactive decision support system for solving real time scheduling problems considering customer and job priorities with schedule interruptions*, Computers Ops Res. **25** (1998), no. 11, 981–995.

[11] J. Blazewicz, W. Domschke, and E. Pesch, *The job shop scheduling problem: Conventional and new solution techniques*, European Journal of Operational Research **93** (1996), 1–33.

[12] R. E. Burkard, T. Fortuna, and C. A.J. Hurkens, *Makespan minimization for chemical batch processes using non-uniform time grids*, Computers and Chemical Engineering **26** (2002), 1321–1332.

[13] F. T.S. Chan and R. Swarnkar, *Ant colony optimization approach to a fuzzy goal programming model for a machine tool selection and operation allocation problem in an FMS*, Robotics and Computer-Integrated Manufacturing **22** (2006), 353–362.

[14] In-Chan Choi and Dae-Sik Choi, *A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups*, Computers & Industrial Engineering **42** (2002), 43–58.

[15] A. R. Ciric and C. A. Floudas, *Heat-exchanger network synthesis without decomposition*, Computers & Chemical Engineering **6** (1991), 385–396.

[16] J. Corominas, A. Espuna, and L. Puigjaner, *Method to incorporate energy integration considerations in multiproduct batch processes*, Comput. Chem. Engng **18** (1994), 1043–1055.

[17] D. Danneberg, T. Tautenhahn, and F. Werner, *A comparison of heuristic algorithms for flow shop scheduling problems with setup times and limited batch size*, Mathematical and Computer Modelling **29** (1999), 101–126.

[18] S. Dunstall and A. Wirth, *A comparison of branch-and-bound algorithms for a family scheduling problem with identical parallel machines*, European Journal of Operational Research **167** (2005), 283–296.

[19] C. A. Floudas and X. Lin, *Continuous-time versus discrete-time approaches for scheduling of chemical processes*, Computers and Chemical Engineering **28** (2004), 2109–2129.

[20] F. Friedler, K. Tarjan, Y. W. Huang, and L. T. Fan, *Combinatorial algorithms for process synthesis*, Comp. Chem. Eng. **16** (1992), S113–S320.

[21] ———, *Graph-theoretic approach to process synthesis: Axioms and theorems*, Chem. Eng. Sci. **47** (1992), 1973–1988.

[22] M. R. Garey and D. R. Johnson, *Computers and intractability: A guide to the theory of NP-completeness.*, New York: W.H. Freeman (1979).

[23] O. A. Ghrayeb, *A bi-criteria optimization: minimizing the integral value and spread of the fuzzy makespan of job shop scheduling problems*, Applied Soft Computing **2/3F** (2003), 197–210.

[24] K. Glismann and G. Gruhn, *Short-term scheduling and recipe optimization of blending processes*, Computers and Chemical Engineering **25** (2001), 627–634.

[25] T. Gonzalez and S. Sahni, *Open shop scheduling to minimize finish time*, Journal of Association for Computing Machinery **23** (1976), 665–679.

[26] R. Grau, A. Espuna, and L. Puigjaner, *Environmental considerations in batch production scheduling*, Computers chem. Engng **19** (1995), S651–S656.

[27] I.E. Grossmann and A.W. Westerberg, *Research challenges in process systems engineering*, AIChE Journal **46** (2000), 1700–1703.

[28] G. Guillén, M. Badell, A. Espuna, and L. Puigjaner, *Simultaneous optimization of process operations and financial decisions to enhance the integrated planning/scheduling of chemical supply chains*, Computers and Chemical Engineering **30** (2006), 421–436.

[29] Z.X. Guo, W.K. Wong, S.Y.S. Leung, J.T. Fan, and S.F. Chan, *Mathematical model and genetic optimization for the job shop scheduling problem in a mixed- and multi-product assembly environment: A case study based on the apparel industry*, Computers & Industrial Engineering **50** (2006), 202–219.

[30] Jin-Kuk Ha, Hyun-Kil Chang, Euy Soo Lee, In-Beum Lee, Beom Sok Lee, and Gyeongbeom Yi, *Intermediate storage tank operation strategies in the production scheduling of multi-product batch processes*, Computers and Chemical Engineering **24** (2000), 1633–1640.

[31] R. Heilmann, *A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags*, European Journal of Operational Research **144** (2003), 348–365.

[32] G. P. Henning and J. Cerda, *A knowledge-based approach to production scheduling for batch processes*, Computers chem. Engng **20** (1996), SI295–SI300.

[33] T. Holczinger, *Módszer köztes tárolókat nem tartalmazó szakaszos müködésü rendszerek ütemezésére*, Ph.D. thesis, Pannon Egyetem (Veszprémi Egyetem), Informatikai Tudományok Doktori Iskola, 2004.

[34] T. Holczinger, J. Romero, L. Puigjaner, and F. Friedler, *Scheduling of multipurpose batch processes with multiple batches of the products*, Hungarian Journal of Industrial Chemistry **30** (2002), 305–312.

[35] S. J. Honkomp, S. Lombardo, O. Rosen, and J. F. Pekny, *The curse of reality - why process scheduling optimization problems are difficult in practice*, Computers and Chemical Engineering **24** (2000), 323–328.

[36] S. J. Honkomp, L. Mockus, and G. V. Reklaitis, *Robust scheduling with processing time uncertainty*, G.V. Reklaitis **21** (1997), S1055–S1060.

[37] ———, *A framework for schedule evaluation with processing uncertainty*, Computers and Chemical Engineering **23** (1999), 595–609.

[38] B. Ivanov, K. Peneva, and N. Bancheva, *Heat integration of batch vessels at fixed time intervals. i. schemes with recycling main fluids*, Hung. J. Ind. Chem **20** (1992), 225–231.

[39] V.K. Jayaraman, B.D. Kulkarni, S. Karale, and P. Shelokar, *Ant colony framework for optimal design and scheduling of batch plants*, Computers and Chemical Engineering **24** (2000), 1901–1912.

[40] I.C Kemp, *Application of the time-dependent cascade analysis in process integration*, Heat Recovery Systems & CLIP **10** (1990), 423–435.

[41] E. Kondili, C. C. Pantelides, and R. W. H. Sargent, *A general algorithm for short-term scheduling of batch operations. I. MILP formulation*, Computers Chem. Engng **17** (1993), 211–227.

[42] B. Lee and G. V. Reklaitis, *Optimal scheduling of cyclic batch processes for heat integration – I. basic formulation*, Computers chem. Engng **19** (1995), 883–905.

[43] ———, *Optimal scheduling of cyclic batch processes for heat integration – II. extended problems*, Computers chem. Engng **19** (1995), 907–931.

[44] J.K. Lenstra and A.H.G. Rirmooy Kan, *Computational complexity of discrete optimization problems*, Annals of Discrete Mathematics **4** (1979), 121–140.

[45] Ching-Fang Liaw, *A hybrid genetic algorithm for the open shop scheduling problem*, European Journal of Operational Research **124** (2000), 28–42.

[46] B. Linnhoff and J. R. Flower, *Synthesis of heat-exchanger networks, part 1, systematic generation of energy optimal networks*, AIChE Journal **24** (1978), 633–642.

[47] B. Linnhoff and E. Hindmarsh, *The pinch design method of heat exchanger networks*, Chemical Engineering Science **38** (1983), 745–763.

[48] T. Majozi, *Heat integration of multipurpose batch plants using a continuous-time framework*, Applied Thermal Engineering **26** (2006), 1369–1377.

[49] T. Majozi and X.X. Zhu (Frank), *A combined fuzzy set theory and MILP approach in integration of planning and scheduling of batch plants–personnel evaluation and allocation*, Computers and Chemical Engineering **29** (2005), 2029–2047.

[50] M. Markowski and K. Urbaniec, *Optimal cleaning schedule for heat exchangers in a heat exchanger network*, Applied Thermal Engineering **25** (2005), 1019–1032.

[51] C. A. Mendez and J. Cerda, *Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities*, Computers and Chemical Engineering **24** (2000), 368–376.

[52] ———, *An MILP continuous-time framework for short-term scheduling of multipurpose batch process under different operation strategies*, Optimization and Engineering **4** (2003), 7–22.

[53] C. A. Mendez, J. Cerda, I. E. Grossmann, I. Harjunkoski, and M. Fahl, *State-of-the-art review of optimization methods for short-term scheduling of batch processes*, Computers and Chemical Engineering **30** (2006), 903–941.

[54] C. A. Mendez, G. P. Henning, and J. Cerda, *An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities*, Computers and Chemical Engineering **25** (2001), 701–711.

[55] M. Mika, G. Waligóra, and J. Weglarz, *Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models*, European Journal of Operational Research **164** (2005), 639–668.

[56] Liu Min and Wu Cheng, *Genetic algorithms for the optimal common due date assignment and the optimal scheduling policy in parallel machine earliness/tardiness scheduling problems*, Robotics and Computer-Integrated Manufacturing **22** (2006), 279–287.

[57] P. Mizsey and E. Rév, *The use of flowsheet simulators in heat exchanger network synthesis*, Hung. J. Ind. Chem. **20** (1992), 91–97.

[58] L. Mockus and G. V. Reklaitis, *A continuous time representation in batch/semicontinuous process scheduling: randomized heuristics approach*, Computers & Chemical Engineering (supp) **20** (1996), S1173–S1177.

[59] ———, *A bayesian approach to batch process scheduling*, Int. Trans. Opl Res. **4** (1997), 55–65.

[60] ———, *Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization*, Computers chem. Engng **21** (1997), no. 10, 1147–1156.

[61] H. P. Nott and P. L. Lee, *An optimal control approach for scheduling mixed batch/continuous process plants with variable cycle time*, Computers and Chemical Engineering **23** (1999), 907–917.

[62] ———, *Sets formulation to schedule mixed batch/continuous process plants with variable cycle time*, Computers and Chemical Engineering **23** (1999), 875–888.

[63] H. Ohta and T. Nakatani, *A heuristic job-shop scheduling algorithm to minimize the total holding cost of completed and in-process products subject to no tardy jobs*, Int. J. Production Economics **101** (2006), 19–29.

[64] S. Orcun, I. K. Altinel, and Ö. Hortacsu, *General continuous time models for production planning and scheduling of batch processing plants: mixed integer linear program formulations and computational issues*, Computers and Chemical Engineering **25** (2001), 371–389.

[65] S. Orcun, A. Discioglu, I. K. AltineL, and Ö. Hortacsu, *Scheduling of batch processes: An industrial application in paint industry*, Computers chem. Engng **21** (1997), S673–S678.

[66] C. C. Pantelides, *Unified frameworks for optimal process planning and scheduling*, Proceedings of the second international conference on foundations of computer-aided process operations (1993), 253–274.

[67] S. A. Papoulias and I. E. Grossmann, *A structural optimization approach in process synthesis-II: heat recovery networks*, Computers & Chemical Engineering **7** (1983), 695–706.

[68] S. Parthasarathy and C. Rajendran, *An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs*, Int. J. Production Economics **49** (1997), 255–263.

[69] J. M. Pinto and I. E. Grossmann, *A continuous time mixed integer linear programming model for short term scheduling of multistage batch plans*, Industrial Engineering Chemical Research **34** (1995), 3037–3051.

[70] ———, *A continuous time MILP model for short term scheduling of batch plants with pre-ordering constraints*, Computers & Chemical Engineering (supp) **20** (1996), S1197–S1202.

[71] L. Puigjaner and A. Espunia, *Prospects for integrated management and control of total sites in the batch manufacturing industry*, Computers chem. Engng **22** (1998), no. 1-2, 87–107.

[72] W. H. M. Raaymakers, J. W. M. Bertrand, and J. C. Fransoo, *Makespan estimation in batch process industries using aggregate resource and job set characteristics*, Int. J. Production Economics **70** (2001), 145–161.

[73] W. H. M. Raaymakers and J. C. Fransoo, *Identification of aggregate resource and job set characteristics for predicting job set makespan in batch process industries*, Int. J. Production Economics **68** (2000), 137–149.

[74] G. Rabadi, M. Mollaghasemi, and G. C. Anagnostopoulos, *A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time*, Computers & Operations Research **31** (2004), 1727–1751.

[75] C. Rajendran and H. Ziegler, *A heuristic for scheduling to minimize the sum of weighted flow time of jobs in a flowshop with sequence-dependent setup times of jobs*, Computers ind Engng **33** (1997), no. 1-2, 281–284.

[76] ———, *Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times*, European Journal of Operational Research **149** (2003), 513–522.

[77] ———, *Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs*, European Journal of Operational Research **155** (2004), 426–438.

[78] M. A. S. S. Ravagnani, A. P. Silva, A. P. Arroyo, and A. A. Constantino, *Heat exchanger network synthesis and optimisation using genetic algorithm*, Applied Thermal Engineering **25** (2005), 1003–1017.

[79] E. Rév and Z. Fonyó, *Diverse pinch concept for heat exchange network synthesis: the case of different heat transfer conditions*, Chem. Eng. Sci. **46** (1991), 1623–1634.

[80] J. Romero, L. Puigjaner, T. Holczinger, and F. Friedler, *Scheduling intermediate storage multipurpose batch plants using the S-graph*, AIChE Journal **50** (2004), no. 2, 403–417.

[81] N. V. Sahinidis and I. E. Grossmann, *MINLP model for cyclic multiproduct scheduling on continuous parallel lines*, Computers and Chemical Engineering **15** (1991), 85–103.

[82] E. Sanmarti, A. Espuna, and L. Puigjaner, *Effects of equipment failure uncertainty in batch production scheduling*, Computers chem. Engng **19** (1995), S565–S570.

[83] E. Sanmarti, A. Espunia, and L. Puigjaner, *Batch production and preventive maintenance scheduling under equipment failure uncertainty*, Computers chem. Engng **21** (1997), l157–1168.

[84] E. Sanmarti, F. Friedler, and L. Puigjaner, *Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation*, Computers chem. Engng **22** (1998), S847–S850.

[85] E. Sanmarti, A. Huercio, A. Espuna, and L. Puigjaner, *A combined scheduling/reactive scheduling strategy to minimize the effect of process operations uncertainty in batch plants*, Computers chem. Engng **20** (1996), S1263–S1268.

[86] E. Sanmarti, L. Puigjaner, T. Holczinger, and F. Friedler, *Combinatorial framework for effective scheduling of multipurpose batch plants*, AIChE Journal **48** (2002), no. 11, 2557–2570.

[87] B. R. Sarker and J. Yu, *Lot-sizing and cyclic scheduling for multiple products in a flow shop*, Computers ind. Engng **30** (1996), no. 4, 799–808.

[88] G. Schilling and C. C. Pantelides, *Optimal periodic scheduling of multipurpose plants in continuous time domain*, Computers and Chemical Engineering **21** (1997), S1191–S1196.

[89] ———, *Optimal periodic scheduling of multipurpose plants*, Computers and Chemical Engineering **23** (1999), 635–665.

[90] R. Smith, *State of the art in process integration*, Applied Thermal Engineering **20** (2000), 1337–1345.

[91] S. K. Stefanis, A. G. Livingston, and E. N. Pistikopoulos, *Environmental impact considerations in the optimal design and scheduling of batch processes*, Computers chem. Engng **21** (1997), 1073–1094.

[92] D.r Subramanian, J. F. Pekny, and G. V. Reklaitis, *A simulation-optimization framework for addressing combinatorial and stochastic aspects of an r&d pipeline management problem*, Computers and Chemical Engineering **24** (2000), 1005–1011.

[93] J. Sun and D. Xue, *A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources*, Computers in Industry **46** (2001), 189–207.

[94] D. Nait Tahar, F. Yalaoui, C. Chu, and L. Amodeo, *A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times*, Int. J. Production Economics **99** (2006), 63–73.

[95] Keah-Choon Tan, R. Narasimhan, P. A. Rubin, and G. L. Ragatz, *A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times*, Omega **28** (2000), 313–326.

[96] L. Tang, H. Xuan, and J. Liu, *A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time*, Computers & Operations Research **33** (2006), 3344–3359.

[97] S. Tzafestas and A. Triantafyllakis, *Deterministic scheduling in computing and manufacturing systems: A survey of models and algorithms*, Journal of Mathematics and Computers in Simulation **35** (1993), 397–434.

[98] N. Vaklieva-Bancheva, B. B. Ivanov, N. Shah, and C. C. Pantelides, *Heat exchanger network design for multipurpose batch plants*, Computers chem. Engng **20** (1996), 989–1001.

[99] H. Yu, A. Reyes, S. Cang, and S. Lloyd, *Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems-part 1. Petri net modelling and heuristic search*, Computers & Industrial Engineering **44** (2003), 527–543.

[100] ———, *Combined petri net modelling and ai based heuristic hybrid search for flexible manufacturing systems-part 2. heuristic hybrid search*, Computers & Industrial Engineering **44** (2003), 545–566.

[101] S. Zdrzałka, *Analysis of approximation algorithms for single-machine scheduling with delivery times and sequence independent batch setup times*, European Journal of Operational Research **80** (1995), 371–380.

[102] X. Zhang and J. F. Bard, *Comparative approaches to equipment scheduling in high volume factories*, Computers & Operations Research **33** (2006), 132–157.

[103] X. Zhang and R. W. H. Sargent, *The optimal operation of mixed production facilities – a general formulation and some approaches for the solution*, Computers & Chemical Engineering **20** (1996), 897–904.