A FOLYAMAT-HÁLÓZATSZINTÉZIS FELADAT KITERJESZTÉSEI

Ph.D. értekezés

Varga József

Témavezető: Dr. Friedler Ferenc

Műszaki informatikai alkalmazások
doktori program

Nagy rendszerek tervezése és irányítása
alprogram

Veszprémi Egyetem
Mérnöki Kar
Számítástudomány Alkalmazása Tanszék

Veszprém
2000

1. A FOLYAMAT-HÁLÓZATSZINTÉZIS FELADAT KITERJESZTÉSEI

Értekezés doktori (PhD) fokozat elnyerése érdekében a Veszprémi Egyetem Műszaki informatikai alkalmazások programja MI3 jelű alprogramjához tartozóan.

Írta:
Varga József

A jelölt a doktori szigorlaton ..... pontot ért el.

Veszprém,

.....
a Szigorlati Bizottság elnöke

Az értekezést bírálóként elfogadásra javaslom (igen/nem).

Első bíráló:

Dr. ..... igen / nem .....
aláírás

Második bíráló:

Dr. ..... igen / nem .....
aláírás

Esetleg harmadik bíráló:

Dr. ..... igen / nem .....
aláírás

A jelölt az értekezés nyilvános vitáján ..... pontot ért el.

A fentiek alapján a doktori oklevél minősítése

Veszprém,

.....
a bíráló bizottság elnöke

2. Tartalomjegyzék

3. Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani mindenkinek, aki lehetővé tette, hogy ez a dolgozat elkészüljön.

Elsősorban témavezetőmnek, Dr. Friedler Ferenc tanszékvezető egyetemi tanárnak a segítségéért és tanácsaiért.

Dr. Hangos Katalin és Dr. L. T. Fan professzoroknak a közös munkánk során több éven keresztül nyújtott segítségükért.

Tanszéki kollégáimnak a segítségükért, a javító szándékú megjegyzéseikért.

Szüleimnek és öcsémnek a biztatásért.

4. Tartalmi összefoglaló

A dolgozatban az alap folyamat-hálózatszintézis feladat gyakorlati szempontból fontos kiterjesztéseinek megoldására alkalmas eszközöket mutatunk be. Az algoritmikus módszerek mindegyike a folyamat-hálózatszintézis feladat megoldására Friedler és munkatársai által kidolgozott kombinatorikus módszerre épül.

A hulladékkezeléssel integrált folyamat-hálózatszintézis feladat definíciója a feladatosztályt leíró feltételeket tartalmazza, ennek megfelelően határoztuk meg a megoldások tulajdonságait. A struktúrákat ebben az esetben az alap folyamat-hálózatszintézis feladattal (a továbbiakban alapeset) megegyezően P-gráffal reprezentáljuk. Megadjuk az optimális megoldás keresési terét meghatározó, a maximális struktúrát generáló (MSGW) algoritmust, illetve bebizonyítjuk az algoritmus helyességét. A branch-and-bound algoritmus szétválasztó lépésének leírásához definiáljuk a döntés leképezés kiterjesztését. Mind az MSGW algoritmus, mind a branch-and-bound algoritmus szétválasztó lépése a feladattípusra általánosan alkalmazható, feladatfüggetlen.

Az integrált folyamat-hálózat- és irányítórendszer tervezés esetén a feladat definíciója bővül, itt is meghatározzuk a megoldásstruktúrákat leíró axiómákat. Ennél a kiterjesztésnél a struktúrákat az ún. CP-gráfokkal reprezentálhatjuk, amelynek a P-gráf az egyik komponense. Ebben az esetben is megadjuk a maximális struktúrát generáló (CMSG) algoritmust és bebizonyítjuk az algoritmus helyességét. A branch-and-bound algoritmus szétválasztó lépése lényegében azonos az eredeti ABB algoritmus szétválasztó lépésével, mindössze egy egyszerű vizsgálatot végzünk minden közbülső részproblémára az irányíthatóságot ellenőrizve.

A szakaszosan folytonos költségfüggvénnyel adott műveleti egységekből álló hálózat szintézise megoldható az ABB algoritmussal is, itt a cél egy a feladattípus speciális tulajdonságait kihasználó algoritmus megadása volt. Elkészült a szakaszosan folytonos költségfüggvénnyel adott műveleti egységekből álló hálózat szintézisét megoldó branch-and-bound algoritmus szétválasztó lépése és az algoritmus definiálásához szükséges részprobléma reprezentáció.

A párhuzamos működési elvű számítógépeken futtatható ABB algoritmus kidolgozásával az algoritmus a manapság egyik leggyorsabban fejlődő számítástechnikai környezetben is használhatóvá válik, az óriási számítási teljesítmény jól kihasználható nagy méretű gyakorlati feladatok megoldásakor.

5. Extensions of Process Network Synthesis (contents)

Algorithmic methods for solving different extensions of process network synthesis (PNS) have been elaborated. These algorithms are based on the accelerated branch-and-bound (ABB) algorithm proposed by Friedler and his colleagues for PNS problems.

For process network synthesis with integrated waste treatment system, (i) the definition of solutions is extended; (ii) the axioms describing the combinatorially feasible structures are defined (iii) the MSGW algorithm for generating the maximal structure to define the search space for the branch-and-bound algorithm solving the problem with proof of correctness is given; (iv) for formal definition of the general branching step of the proposed branch-and-bound algorithm an extended decision-mapping has been introduced. Both the MSGW algorithm and the branching step of the branch-and-bound method are general; they are independent of the actual problem.

In the case of integrated process and control system synthesis, (i) the definition of solutions is extended; (ii) the set of axioms describing the properties of combinatorially feasible and controllable (CFC) solution structures are given; (iii) an algorithm for generating the maximal CFC structure is presented followed by the proof of correctness; (iv) the branching step of the ABB algorithm is extended by a procedure for checking the controllability of a partial problem.

Although the ABB algorithm can solve a PNS problem including operating units with sectionally continuous cost functions, an efficient algorithm exploiting the special properties of such problems is developed. An extended branching method and a new representation of partial problems are elaborated.

The parallelization of the ABB algorithm allows us to utilize the acceleration achieved by the usage of multiple processors besides the acceleration of the sequential algorithm compared to traditional methods.

6. Extensionen der Produktionsnetzwerksynthese (Inhalt)

Algorithmische Methoden für das Lösen unterschiedlicher Erweiterungen der Produktionsnetzwerksynthese (PNS) sind ausgearbeitet worden. Diese Algorithmen basieren auf dem beschleunigten Verzweigungsalgorithmus (ABB), der von Friedler und seinen Mitarbeitern für PNS-Probleme vorgeschlagen wurde.

Bei synthetisierten Prozeßnetzen aus integriertem Abfallbehandlungssystem, ist die Definition der Lösungen erweitert und dergleichen ist das Set der Axiome, welches die Eigenschaften der kombinatorisch möglichen Lösungsstrukturen beschreibt. Die Lösungsstrukturen werden durch P-Diagramme wie im Fall von den PNS-Problemen, dargestellt. Die Superstruktur definiert den Suchraum des Algorithmus, der das Problem löst. Der MSGW-Algorithmus legt die Superstruktur fest; die Korrektheit dieses Algorithmus wurde nachgewiesen. Für formale Definition des allgemeinen erweiterten Jobsteps des vorgeschlagenen Verzweigungsalgorithmus, wurde ein erweitertes Entscheidungs-mapping eingeführt; es kann Abfallbehandlungssysteme behandeln. Beide Algorithmen, der MSGW und der erweiterte Jobstep der Verzweigungsmethode, sind allgemein anwendbar; sie sind vom tatsächlichen Problem unabhängig.

Im Fall integrierter Prozeß- und Steuersystemsynthese, wird die Definition der Lösungen auch ausgedehnt und dasselbe gilt für das Set an Axiomen, welche die Eigenschaften der kombinatorisch möglichen und kontrollierbaren, Lösungsstrukturen beschreibt (CFC). Die Lösungsstrukturen werden von CP-GRAPHEN beschrieben, das P-Diagramm ist ein Bestandteil des CP-GRAPHEN. Ein Algorithmus für das Festlegen der maximalen CFC-Struktur, gefolgt vom Beweis von Korrektheit, wird dargestellt. Der erweiterte Jobstep des ABB-Algorithmus wird durch ein Verfahren für die Überprüfung der Steuerbarkeit eines partiellen Problems ausgedehnt. Obgleich der ABB-Algorithmus ein PNS-Problem einschließlich der funktionierenden Maßeinheiten mit geschnitten ununterbrochenen Kostenfunktionen lösen kann, wird ein noch leistungsfähiger Algorithmus, der die speziellen Eigenschaften solcher Probleme ausnutzt, entwickelt. Eine erweiterte Branching-Methode und eine neue Darstellung der Teilprobleme werden ausgearbeitet.

Die Paralellität des ABB-Algorithmus erlaubt es, die Beschleunigung derart zu steigern, daß der Gebrauch von Mehrfachprozessoren zu zusätzlicher Zeiteinsparung führt, die in der Arbeit mit traditionellen Methoden verglichen wird.

7. Bevezetés

A dolgozat a Friedler és munkatársai által kidolgozott folyamat-hálózatszintézis feladatot megoldó kombinatorikus módszer, és az erre épülő gyorsított branch-and-bound algoritmus [16], az ABB algoritmus, lehetséges továbbfejlesztéseivel foglalkozik. A továbbiakban ezt a módszert tekintjük alapnak és az ABB algoritmussal megoldható feladatosztályt a folyamat-hálózatszintézis alapfeladatának.

A folyamat-hálózatszintézis feladat rövid leírása: egy folyamat-hálózatszintézis feladat definiálásakor meg kell adnunk a lehetséges építőelemek (a továbbiakban **műveleti egységnek** hívjuk) halmazát, melyek valamiből (input) valamit (output) gyártanak, a műveleti egységek működését leíró összefüggéseket (inputok, outputok, a műveleti egység lehetséges állapotai, költségfüggvénye, ezek kapcsolata), a rendelkezésre álló nyersanyagok halmazát, az ezekre vonatkozó esetleges korlátokat, az előállítani kívánt termékéket és jellemzőiket, továbbá meg kell határoznunk, hogy milyen szempontból keressük az optimális megoldást (azaz meg kell adnunk egy költségfüggvényt).

A cél műveleti egységek egy halmazának (a továbbiakban **struktúra**) kiválasztása és ezek működését leíró paraméterek megadása úgy, hogy a struktúrát ezen paramétereknek megfelelően működtetve kapott **hálózat** (a feladat megoldása) megfelel a feladat definíciójában adott követelményeknek és a szintén definiált szempontból optimális.

A folyamat-hálózatszintézis feladatot megoldó módszerek célja tehát egy optimális hálózat megkeresése, amely egy struktúrának és a struktúra optimális működésének a meghatározását jelenti.

A dolgozatban az irodalmi áttekintés után a Friedler és munkatársai által kidolgozott gyorsított branch-and-bound algoritmust külön fejezetben mutatjuk

be, mivel az szolgál a dolgozatban ismertetett kiterjesztések alapjául. Ezt követi a kiterjesztések megoldására kidolgozott módszerek bemutatása.

A folyamat-hálózatszintézis feladat kiterjesztésén olyan feladatot értünk, amelyben az alapfeladathoz képest további feltételek adottak. Az itt bemutatott kiterjesztések mellett természetesen további kiterjesztések is léteznek, valamint a folyamat-hálózatszintézis feladat kiterjesztései általában szabadon kombinálhatóak.

A dolgozatban az alábbi kiterjesztéseket mutatjuk be: hulladékkezeléssel integrált folyamat-hálózatszintézis, integrált folyamat-hálózat- és irányítórendszer tervezés, folyamat-hálózatszintézis szakaszosan folytonos célfüggvényű műveleti egységekkel, folyamat-hálózatszintézis feladat megoldása párhuzamos működési elvű számítógépeken.

Hulladékkezeléssel integrált folyamat-hálózatszintézis esetén a feladat definíciója összetettebb, az alapfeladatban is definiált szükséges termékek halmaza mellett a nem megengedett outputok halmaza is adott. A kiterjesztés legfontosabb alkalmazási területe a vegyipar, ugyanakkor a kidolgozott módszer általánosan alkalmazható műszaki termelő rendszereknél. A megengedett, de nem kívánatos outputokhoz a környezet terhelésével arányos értékeket rendelve és ezeket a célfüggvénybe építve olyan módszert kapunk, amely az optimális megoldást a költségen kívül (esetleg helyett) más szempontokat is figyelembe véve határozza meg.

Az integrált folyamat-hálózat- és irányítórendszer tervezés esetén a feladat definíciója szintén tartalmaz további feltételeket, itt irányíthatósági szempontokat veszünk figyelembe az optimális megoldás keresésekor. Az irányítás a működés során előforduló dinamikus változásokhoz kapcsolódik, ugyanakkor a folyamat-hálózatszintézis feladatot megoldó módszerek célja egy optimális hálózat megkeresése, amely egy struktúra és annak egy optimális - statikus jellegű - működésének a meghatározását jelenti. Így egyszerű Boolean típusú irányíthatósági szempontokat vehetünk figyelembe, az alkalmazhatóság (megfigyelhetőség, szabályozás) szempontjából már így is lényegesen jobb megoldást kaphatunk.

Szakaszosan folytonos költségfüggvény alkalmazása a folyamat-hálózatszintézis feladatban a matematikai modell specializálását jelenti. Ez a feladatosztály megoldható az alap ABB algoritmussal is megfelelő korlátozó függvény alkalmazásával, viszont az így kapott módszer az óriási számításigény miatt csak korlátozottan lenne használható. Az itt bemutatott módszer ezt a feladatot oldja meg az ABB algoritmusba egy összetettebb szétválasztó lépést integrálva.

A gyorsított branch-and-bound algoritmus párhuzamos működési elvű számítógépeken futtatható változata az eredeti szekvenciális algoritmus módosítása, lehetővé téve még nagyobb méretű folyamat-hálózatszintézis feladatok gyors megoldását, feltéve, hogy a megfelelő eszköz rendelkezésre áll.

8. Irodalmi áttekintés

A fejezetben először röviden ismertetünk néhány ismert, a folyamat-hálózatszintézis feladat megoldására kidolgozott általános módszert. Ezt követően a dolgozatban ismertetett folyamat-hálózatszintézis feladat kiterjesztésekkel azonos, vagy ahhoz hasonló feladatosztályt megoldó módszereket mutatunk be.

Mivel a dolgozatban ismertetett módszerek mindegyike a branch-and-bound technikára épül, a szakirodalomban széles körben publikált módszer egy formális leírását a dolgozat végén függelékként adjuk meg.

8.1. Folyamat-hálózatszintézis feladatot megoldó általános módszerek

A folyamat-hálózatszintézis feladat megoldására kidolgozott módszerek egy része heurisztikus szabályokat alkalmaz, itt az optimális megoldás megtalálása nem garantált. A korábban kidolgozott egzakt matematikai módszerekre épülő eljárások nagy része egy általános matematikai programozási módszert alkalmazott a folyamat-hálózatszintézis feladat megoldására [5, 7, 25, 40], ami a folyamat-hálózatszintézis kombinatorikus jellegének köszönhetően egy vegyes egész, sok bináris változót tartalmazó matematikai programozási feladat megoldását jelenti a módszertől függetlenül, például Benders dekompozíció [25], külső közelítés [5]. Egy ipari méretű feladat megoldása óriási számítási igényű, ezek a módszerek nem használják ki a folyamat-hálózatszintézis feladat jellegzetességeit, lényegében a feladat matematikai modelljének felírása után nincs szerepe az eredeti feladatnak. A módszerek némelyike megengedi heurisztikus algoritmusok alkalmazását a számítások gyorsítása érdekében, ez viszont a globális optimum figyelmen kívül hagyását jelentheti a megoldás keresésekor.

Kifejezetten folyamat-hálózatszintézis feladatok megoldására Grossmann és munkatársai [34] dolgoztak ki egy módszert, amely az optimális megoldás keresésekor a folyamat-hálózatszintézis feladat kombinatorikus tulajdonságait leíró logikai összefüggéseket figyelembevéve teszi hatékonyabbá az optimális megoldás keresését. Brendel és munkatársai [2] bebizonyították, a gráfalgoritmusokra alapozó kombinatorikus technika megfelelő alapokat szolgáltat a módszerhez, azaz levezethető belőle. A módszer hátránya, hogy a logikai formulákat és a matematikai programozási módszereket együttesen alkalmazó módszer hatékony megvalósítása nehézkes (például milyen programnyelvet használjunk), továbbá a publikált módszer nem megfelelően kezel bizonyos eseteket (például az előforduló köröket (recirkulációkat) megszűnteti, így lehetséges megoldásokat nem vesz figyelembe a megoldás keresése során).

Kombinatorikus módszereket, részben a branch-and-bound technikát, részben dinamikus programozást alkalmaz Fraga és McKinnon [8, 9]. Módszerük azonban nem törekszik az általános alkalmazhatóságra, a feladatot részproblémákra bontó szétválasztó lépés feladatfüggő a módszerben. A szétválasztás folytonos változók szerint is lehetséges a módszerben, ekkor a folytonos változót diszkrét értékek egy véges halmazával közelítik.

A folyamat-hálózatszintézis feladat megoldására kidolgozott ABB algoritmust a következő fejezetben ismertetjük.

8.2. Hulladékkezeléssel integrált folyamat-hálózatszintézis

A hulladékkezeléssel integrált folyamat-hálózatszintézis megoldására kidolgozott hagyományos módszerek közös jellemzője, hogy a kívánt terméket termelő hálózat tervezését és a tiltott, illetve magas költséggel járó outputok kezelését végző hálózat tervezését két külön lépésben oldják meg. Ez azonban általában nem vezet optimális vagy közel optimális megoldáshoz.

Az amerikai EPA (Environmental Protection Agency, US) által definiált hierarchiában [1, 35] a hulladékkezeléssel integrált folyamat-hálózatszintézis feladat megoldásában elsődleges szerepet kap a felhasznált nyersanyagok

mennyiségének csökkentése (ami természetesen a szükséges termékek mellett kevesebb egyéb outputot eredményez), második helyen az újrahasznosítás szerepel, a hulladékkezelő rendszer építése csak ezeket követi. Míg az első két lehetőséget, ami lényegében a termelő hálózat paramétereinek meghatározásakor a hulladékminimalizálás figyelembevételét jelenti, több módszerben is a termelő hálózat tervezésével integrálva alkalmazzák, a hulladékkezelő és termelő hálózat valódi integrált tervezésére eddig egyetlen módszer sem vállalkozott.

Több hulladékkezeléssel foglalkozó módszer [3, 4] a fenti hierarchiának csak az első, vagy első két lépését alkalmazza, azaz a hulladékkezelő hálózat tervezése nem is része a módszernek. A Crabtree és El-Halwagi által javasolt módszer [4] kész termelő hálózat hulladékkibocsátását minimalizálja a már fix struktúra állapotváltozóinak módosításával, de a termelő rendszer struktúráját nem módosítja, azaz nem általános integrált tervező módszert ad, a termelő hálózat tervezésétől elválasztja a hulladékkezelést.

A Berger által javasolt hulladékkezeléssel integrált folyamat-hálózatszintézis feladatot megoldó módszer [1] az optimális megoldás tervezését több lépésre bontja (stage gate model). Az első lépésben a működést leíró paraméterek meghatározásakor figyelembe veszi a hierarchia első két szintjét. Viszont a hulladékkezelő rendszer tervezését egy különálló lépésben oldja meg. A módszer nyilván jobb megoldást eredményez, mint a szükséges termékek termelését és a nem megengedett outputok kezelését végző hálózatokat teljesen elkülönülten tervező módszerek, de az integráció itt sem teljes.

8.3. Integrált folyamat-hálózat- és irányítórendszer tervezés

A hagyományos tervezési módszer szerint egy hálózat és irányítórendszerének tervezése két egymást követő lépésben történik. A legtöbb munka az irányítórendszer tervezésével kapcsolatban a kiértékelésre összpontosít, csak kevesen adnak algoritmikus eljárásokat, amelyek az irányíthatóság mértékét is integrálják a tervezési lépésbe [37].

Nishida és munkatársai [30, 31] törekedtek elsőként az irányíthatóság szisztematikus figyelembevételére folyamat-hálózatszintézis feladatok megoldásakor. Nagyon kevés publikáció jelent meg, amelyben olyan módszert ismertetnek, amely egyidejűleg törekszik a gazdaságos és az irányítható tervezésre [20, 28, 30, 31], ezek speciális folyamat-hálózatszintézis feladatok irányítórendszerrel integrált tervezésére adnak megoldást általános matematikai programozási módszereket alkalmazva.

8.4. Szakaszosan folytonos költségfüggvény alkalmazása folyamat-hálózatszintézis feladatban

A feladattípus az alap folyamat-hálózatszintézis feladathoz hasonlóan megoldható általános matematikai programozási módszerekkel, viszont a szakaszon folytonos függvények jelenléte miatt ez különösen nehéz, időigényes, nagy ipari feladatok megoldása szinte lehetetlen.

A feladattípus megoldására alkalmas a Raman és Grossmann által kidolgozott matematikai logikát használó módszer, az úgynevezett diszjunktív programozás, amellyel a különböző méretű műveleti egységek használata egy-egy megoldásban logikai formulákkal jól leírható. Ez a módszer jóval hatékonyabb, mint egy általános matematikai programozási módszer alkalmazása, azonban néhány problémát megoldatlanul hagyott: például a műveleti egységek kapacitásaira felső korlát algoritmikus számolása hiányzik, ezt inputként várja a módszer.

8.5. Folyamat-hálózatszintézis feladat megoldása párhuzamos működési elvű számítógépekkel

A szakirodalomban fellelhető módszerek nagy része [23, 26] általános matematikai programozási módszerek párhuzamos működési elvű számítógépekre adaptált változatát [29, 32, 36] alkalmazza folyamat-hálózatszintézis feladat megoldására.

A folyamat-hálózatszintézis feladat megoldására kidolgozott kombinatorikus módszer párhuzamos feldolgozásra alkalmas változatát készítette el Fraga és McKinnon [9], az átalakított módszer dinamikus programozást használ. A

módszert transzputeren, 64 processzoros Intel i860-as számítógépen és munkaállomás hálózaton valósították meg. Elkészítették mind a dinamikus processzorterhelés-kiegyenlítéssel (load balance) dolgozó, mind a hierarchikus mester-szolga (master-slave) alapú változatot. Mindkét típusú algoritmussal, minden géptípus esetén közel lineáris gyorsulást tapasztaltak, sőt a feladatok méretének növelésével a gyorsulás tovább javult. A mester-szolga algoritmus alkalmazásakor a mester nem vált szűk keresztmetszetté a feladatok megoldása során.

9. Folyamat-hálózatszintézis kombinatorikus módszerrel

A folyamat-hálózatszintézis kiterjesztéseinek megoldására javasolt módszerek a Friedler és munkatársai által kidolgozott, gráfalgoritmusokra alapozó, kombinatorikus technikára épülnek [16].

A folyamat-hálózatszintézis alapfeladatának formális definíciója: adott **műveleti egység**ek egy halmaza, minden műveleti egység egy (in_i, out_i, m_i, k_i) rendezett négyesnek tekinthető, ahol in_i az i. műveleti egység inputjainak halmaza; out_i az i. műveleti egység outputjainak halmaza; m_i az i. műveleti egység működését leíró függvény; a műveleti egység inputjainak és egyéb állapotváltozóinak függvényében megadja a műveleti egység outputjait; a k_i az i. műveleti egység költségfüggvénye.

A műveleti egységek közötti kapcsolatot a műveleti egységek inputjai és outputjai jelentik. Ezeket a továbbiakban ***anyag***oknak nevezzük, bár az input vagy output lehet energia, vagy például pozíció is egy szállítási feladat esetén. Az anyagok halmaza természetesen külön is definiálható, bár egyetlen műveleti egységhez sem tartozó anyag definiálása nyilván felesleges. ***Hálózaton*** a műveleti egységek egy részhalmazának valamilyen állapotváltozók szerinti működtetését értjük, amely bizonyos input anyagokból bizonyos output anyagokat állít elő. A ***hálózat struktúráját*** definiálhatjuk műveleti egységek halmazának állapotváltozók nélküli megadásával.

Ha a lehetséges input-output anyagok közül bizonyos anyagok (***nyersanyagok***) rendelkezésünkre állnak (korlátozott vagy korlátlan mennyiségben) és célunk valamely más anyaghalmaz (***termékek***) előállítása (alulról korlátozott mennyiségben), akkor a folyamat-hálózatszintézis alapfeladata a műveleti egységek egy olyan részhalmazának megkeresése a megfelelő állapotváltozókkal, amelyekből felépülő hálózat (megoldás) az adott inputok (nyersanyagok) segítségével a kért outputokat (termékek) minimális költséggel

állítja elő. A hálózat költsége általában a műveleti egységek, a nyersanyagok, a termékek, és az egyéb (hasznos, vagy nem kívánt) outputok költségéből számítható. Ez a hálózat lesz a folyamat-hálózatszintézis feladat optimális megoldása. A folyamat-hálózatszintézis alapfeladata tehát definiálható a műveleti egységek, nyersanyagok és termékek megadásával.

Mint a feladat definíciójából is látható, a folyamat-hálózatszintézis részben kombinatorikus (műveleti egységek egy részhalmazának keresése), részben matematikai programozás (költségfüggvény, állapotváltozók) jellegű feladat, ami egyben megoldását is nehézzé teszi, hiszen a kombinatorikus jelleg miatt a feladat matematikai modellje egy sok bináris változót tartalmazó vegyes egész programozási feladat lesz.

A megoldások mindegyikének rendelkeznie kell néhány triviális tulajdonsággal, viszont ezek felírása a matematikai modellbe azt lényegesen bonyolultabbá tenné, azaz a megoldás keresésének hatékonysága nem javulna, így ezek a tulajdonságok csak a matematikai modellel dolgozva nem használhatóak ki.

A gyorsított branch-and-bound algoritmus lényegében egy olyan speciális algoritmus, amely a folyamat-hálózatszintézis feladat kombinatorikus jellegét kihasználva több nagyságrenddel gyorsítja az optimális megoldás keresését az általános matematikai programozási módszerekhez képest.

A kombinatorikus jelleg kihasználása nyilvánvalóan azt jelenti, hogy a módszer nem egyszerűen egy matematikai modellt old meg, hanem a hálózatok struktúráival is dolgozik a keresés során, ezért alapvető fontosságú, hogy a struktúrareprezentáció egyértelmű, matematikai értelemben szigorú, és (kombinatorikus) algoritmusokkal jól kezelhető legyen.

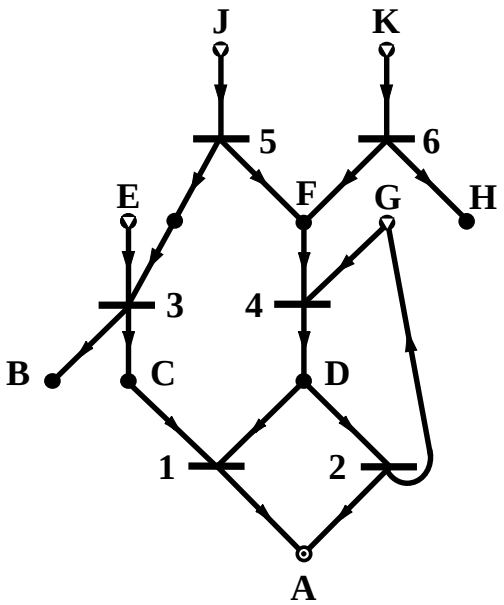
9.1. Struktúra reprezentáció

Bevett szokás, hogy a hálózat struktúráját egy irányított gráffal írják le, de ez nem alkalmas az egyértelmű megadására, ezért a hálózat struktúráját egy irányított páros gráffal, az úgynevezett **P-gráffal** reprezentáljuk [10, 11, 17]. A gráf csúcspontjait a folyamat-hálózatszintézis feladat definíciójában adott műveleti egységek és a hozzájuk kapcsolódó anyagok alkotják. A gráf élei az

anyagok és műveleti egységek közötti kapcsolatok: egy *b* műveleti egység típusú csúcsból él vezet egy *a* anyag típusú csúcshoz, ha *a* eleme *b* outputhalmazának ($a \in out_b$), illetve egy *a* anyag típusú csúcsból él vezet egy *b* műveleti egység típusú csúcshoz, ha *a* eleme *b* inputhalmazának ($a \in in_b$). Az ily módon definiált gráf nyilván páros, hiszen soha nincs él két azonos típusú csúcs között. A P-gráfot a csúcsokat alkotó anyag- és műveleti egység halmazból álló párossal adhatjuk meg, például (M, O) P-gráf, az éleket a műveleti egységek egyértelműen meghatározzák.

A P-gráf ábrázolásakor a műveleti egység típusú csúcsokat vízszintes vonallal (▬), az anyag típusú csúcsokat körrel (●) jelöljük. Az anyag típusú csúcsok között a nyersanyagokat ▽, a termékeket ⊙ különbözteti meg a többi anyagtól.

1. példa. Az 1. ábra egy 6 műveleti egységből álló folyamat-hálózatszintézis feladat P-gráfját szemlélteti, és mint az az ábráról is leolvasható, a feladat az 1, 2,..., 6 műveleti egységek valamely kombinációjával az A anyag termelése úgy, hogy nyersanyagként az E, G, J és K anyagok állnak rendelkezésre.



1. ábra. Az ({A, B, C, D, E, F, G, H, J, K}, {1, 2, 3, 4, 5, 6}) P-gráf.

9.2. Kombinatorikusan lehetséges struktúrák

Az optimális megoldás struktúrájának rendelkeznie kell néhány olyan tulajdonsággal, amelyek függetlenek a műveleti egységek matematikai modelljétől. Az optimális megoldás keresése során ezeket a kombinatorikus tulajdonságokat figyelembevéve nagyságrendekkel javítható a keresés hatékonysága. Ezeket a nyilvánvaló tulajdonságokat mint axiómákat fogalmazzuk meg [11]:

(S1) Minden termék szerepel a struktúrában.

(S2) Egy a struktúrában szereplő anyag akkor és csak akkor nyersanyag, ha egyetlen a struktúrában szereplő műveleti egység sem állítja elő.

(S3) Minden a struktúrában szereplő műveleti egység a folyamat-hálózatszintézis feladatban definiált.

(S4) Minden a struktúrában szereplő műveleti egységtől vezet út termékhez.

(S5) Ha egy *a* anyag része a struktúrának, akkor létezik a struktúrában olyan műveleti egység, amelynek *a* inputja vagy outputja.

Az S4 axiómában szereplő út a gráfelméletben szokásos irányított utat jelenti a struktúra P-gráfjában.

A megoldások struktúráinak, a továbbiakban megoldásstruktúráknak, nyilván rendelkezniük kell a fenti tulajdonságokkal, de ez csak szükséges feltétel, nem elégséges. Elképzelhető, hogy a fenti feltételeket teljesítő struktúra nem működtethető úgy, hogy a kívánt mennyiségű terméket előállítsa a rendelkezésre álló nyersanyagokból. Azokat a struktúrákat, amelyek teljesítik a fenti axiómákat **kombinatorikusan lehetséges struktúráknak** nevezzük. Az optimális megoldás keresését a kombinatorikusan lehetséges struktúrák halmazára szűkítve, a keresési tér lényegesen csökken. Például egy 35 műveleti egységgel definiált folyamat-hálózatszintézis feladat esetén [17] 2^{35}-1, azaz több mint 34 milliárd struktúra adható meg, ugyanakkor a kombinatorikusan lehetséges struktúrák száma mindössze 3465.

Láthattuk, hogy a kombinatorikusan lehetséges struktúrák száma általában nagyságrendekkel kisebb, mint a berendezések halmazának összes lehetséges

részhalmaza, ugyanakkor a műveleti egységek számának növelésével ez a szám is exponenciálisan nőhet. A teljes leszámlálás, azaz az összes kombinatorikusan lehetséges struktúra generálása, majd a fix struktúra matematikai modelljének megoldása (amely már nem tartalmazza a kombinatorikus jellegből adódó bináris változókat, a műveleti egységek modelljéből írható fel, általában egy LP vagy NLP feladat) nem megfelelő módszer. Az ilyen jellegű feladatok megoldására az egyik legelterjedtebb módszer a branch-and-bound (Függelék) alkalmazása. A branch-and-bound fontosabb előnyei a leszámlálásnál lényegesen hatékonyabb keresésen túl: (i) az optimális megoldás mellett az n legjobb megoldás is generálható, (ii) heurisztikus algoritmusokkal kombinálható, (iii) alkalmas párhuzamos feldolgozásra.

9.3. Maximális struktúra algoritmikus generálása

A keresés terét minimalizálhatjuk, ha csak azokat a műveleti egységeket vesszük figyelembe a keresés során, amelyek valamely kombinatorikusan lehetséges struktúra részei lehetnek. Mivel a kombinatorikusan lehetséges struktúrák halmaza véges és zárt az unióra [12], ha ez a halmaz nem üres, akkor létezik **maximális struktúra**, melynek minden kombinatorikusan lehetséges struktúra részhalmaza. Így a folyamat-hálózatszintézis feladat megoldására készített algoritmusban erre a struktúrára szorítkozhatunk a keresés során.

A Friedler és munkatársai által kidolgozott MSG algoritmus ezt a maximális struktúrát generálja az összes kombinatorikusan lehetséges struktúra generálása nélkül polinomiális időben [12]. Az MSG algoritmus több, a dolgozatban ismertetett folyamat-hálózatszintézis feladat kiterjesztés esetén az adott feladatosztály megoldásához szükséges maximális struktúra generálásához szolgál alapként.

9.4. Döntés leképezés

Az MSG algoritmus a minimális komplexitást biztosító keresési teret határozza meg. A folyamat-hálózatszintézis feladatot megoldó branch-and-bound algoritmus pontos megadásához definiálni kell a korlátozó (bounding) és a szétválasztó (branching) lépéseket. A korlátozó eljárás a folyamat-

hálózatszintézis feladat matematikai modelljétől függ, ezért általánosan nem adható meg, a MINLP vagy MILP matematikai programozási feladatok megoldásakor az általános branch-and-bound módszereknél szokásos módon célszerű a relaxált modellt definiálni. A szétválasztó lépésnél is alkalmazhatóak az általános módszerek, ebben az esetben azonban a folyamat-hálózatszintézis feladat speciális kombinatorikus tulajdonságait nem használjuk ki. A folyamat-hálózatszintézis feladat megoldására kidolgozott ABB algoritmus ezeket a kombinatorikus tulajdonságokat használja ki a szétválasztó lépésben. A szétválasztó lépés alapja a döntés leképezés [14], amely egyrészt garantálja a branch-and-bound algoritmusok szétválasztó lépésére megkívánt tulajdonságok teljesülését, másrészt segítségével a szétválasztó lépések sorrendjét úgy határozhatjuk meg, hogy az minimális számú részprobléma megoldását tegye szükségessé. Az alábbiakban a döntés leképezés alapvető definícióit és tulajdonságait mutatjuk be. Mivel az alábbi állítások a folyamat-hálózatszintézis feladat matematikai modelljétől függetlenek, csak a megoldásstruktúrákra vonatkoznak, így a folyamat-hálózatszintézis feladat megadására megfelel a feladat (M, O) P-gráfja.

Jelölje Δ azt a leképezést az anyagok és a műveleti egységek hatványhalmaza között, amely minden $X \in M$ anyagra megadja az X-et előállító műveleti egységek halmazát, azaz $\Delta(X) = \{i \in O \mid X \in out_i\}$.

1. definíció. Legyen $m \subseteq M$ és $\delta(X) \subseteq \Delta(X)$ minden $X \in m$-re. Ekkor δ-t, mint leképezést az m halmazból a műveleti egységek részhalmazainak halmazába, $\delta[m] = \{(X, \delta(X)) \mid X \in m\}$, **döntés leképezésnek** nevezzük m felett.

2. definíció. A $\delta[m]$ **döntés leképezés komplementere** a $\overline{\delta}[m] = \{(X, \Delta(X) \setminus \delta(X)) \mid X \in m\}$ leképezés.

3. definíció. A $\delta[m]$ döntés leképezés ($m \neq \emptyset$) akkor és csak akkor **konzisztens**, ha minden $X, Y \in$ m-re $\delta(X) \cap \overline{\delta}(Y) = \emptyset$.

Jelölje $op(\delta[m])$ a $\delta[m]$ döntés leképezés műveleti egységeit, azaz $op(\delta[m]) = \{o \in O \mid o \in \delta(X), X \in m\}$, továbbá $mat(o) = \{x \in M \mid$ valamely $u \in o$ műveleti egységre $x \in in_u$ vagy $x \in out_u\}$.

4. definíció. Legyen $\delta[m]$ egy konzisztens döntés leképezés, $\mathbf{o} = op(\delta[m])$, $\mathbf{m} = mat(\mathbf{o}) \cup m$, és $\delta'[\mathbf{m}] = \{(X, Y) \mid X \in \mathbf{m}$ és $Y = \{i \in \mathbf{o} \mid X \in out_i\}\}$. Ekkor $\delta'[\mathbf{m}]$

döntés leképezés a δ[m] **döntés leképezés lezártja**. A δ[m] döntés leképezés zárt, ha δ[m] = δ'[**m**].

5. definíció. Két konzisztens döntés leképezés **ekvivalens**, ha a lezártjuk azonos.

A szükséges fogalmak bevezetése után megadhatjuk a döntés leképezés és a P-gráf kapcsolatát.

6. definíció. Adott az (m, o) P-gráf. Ha egy $m' \subseteq m$ halmazra teljesül, hogy minden $i \in o$-ra $out_i \cap m' \neq \emptyset$, akkor m'-t az (m, o) **P-gráf aktív halmazának** nevezzük.

7. definíció. Legyen m' az (m, o) P-gráf aktív halmaza. A $\delta[m'] = \{(X, Y) \mid X \in m' \text{ és } Y = \{i \in o \mid X \in out_i\}\}$ döntés leképezés az (m, o) **P-gráfhoz tartozó döntés leképezés**.

Minden P-gráfhoz tartozó döntés leképezés konzisztens. A fordított kapcsolat bemutatásához tekintsük a δ[m'] konzisztens döntés leképezést. Legyen o = op(δ[m']), és m = mat(o)∪m'. Ekkor (i) az (m, o) P-gráf; (ii) m' az (m, o) P-gráf aktív halmaza; (iii) δ[m'] az (m, o) P-gráfhoz tartozó döntés leképezés. Ez alapján:

8. definíció. Egy δ[m'] **konzisztens döntés leképezés P-gráfja** (m, o), ahol o = op(δ[m']), és m = mat(o)∪m', és gráf(δ[m'])-mel jelöljük.

9. definíció. Legyenek $\delta_1[m_1]$ és $\delta_2[m_2]$ konzisztens döntés leképezések. A $\delta_1[m_1]$ a $\delta_2[m_2]$ **kiterjesztése**, $\delta_1[m_1] \geq \delta_2[m_2]$, ha $m_1 \supseteq m_2$ és $\delta_1(X) = \delta_2(X)$ minden $X \in m_2$-re. A kiterjesztés reláció parciális rendezés a konzisztens döntés leképezések halmazán.

9.5. ABB algoritmus

A szükséges előkészületi lépések után definiálhatjuk a gyorsított branch-and-bound algoritmust. Mivel a célfüggvény és a műveleti egységek modellje feladatfüggő, az általános branch-and-bound leírásnál (Függelék) adott módon használjuk az f, F és G függvényeket, ezek azonban a kombinatorikus módszerek leírásához nem szükségesek.

A folyamat-hálózatszintézis feladat kombinatorikus jellegéből adódóan a feladat matematikai modellje vegyes-egész programozási feladat: Tegyük fel, hogy adott egy folyamat-hálózatszintézis feladat, amelynek maximális struktúrája n

műveleti egységet tartalmaz (a feladat definíciójában esetlegesen szereplő többi műveleti egység elhagyható). Minden o_i műveleti egységhez hozzárendelve egy y_i bináris változót, a megoldások műveleti egységek modelljétől független struktúrái, azaz a maximális struktúra részgráfjai, az (y_1, y_2, ..., y_n) vektor alapján egyértelműen meghatározhatóak.

A folyamat-hálózatszintézis feladat matematikai modelljében a további változók a műveleti egységek állapotváltozói, ezek általában folytonos változók. A folytonos változók miatt a folyamat-hálózatszintézis feladat lehetséges megoldásainak halmaza általában nem véges, viszont a tervezés szempontjából elegendő, ha a struktúrájukban eltérő megoldásokat tekintjük csak különbözőnek.

Az adott megoldásstruktúra optimális működésének meghatározása egy jól elkülöníthető feladat, amely az adott struktúra költségére korlátot számoló függvény feladata. Ennek megfelelően a gyorsított branch-and-bound algoritmus esetén a G korlátozó függvényről feltételezzük, hogy egy olyan P_i részproblémára, amely struktúrájukban azonos megoldások halmaza, a $G(P_i) = F(P_i)$ összefüggés fennáll.

Mivel a kombinatorikusan lehetséges megoldások halmaza véges, ez a feltétel elégséges a gyorsított branch-and-bound algoritmus megállási feltételeként is, továbbá egy P_i részprobléma definiálásakor elegendő megadni, mely kombinatorikusan lehetséges struktúrákat tartalmazza. Így például a kezdő P_0 részprobléma az összes kombinatorikusan lehetséges megoldásstruktúrát magában foglalja.

A hagyományos branch-and-bound algoritmusok szétválasztó lépése gyakran úgy bontja fel a részproblémákat, hogy az a kombinatorikusan lehetséges struktúrák szerint nem jelent szétválasztást. A gyorsított branch-and-bound algoritmus szétválasztó lépése tér el lényegesen a "hagyományos" vegyes egész típusú matematikai programozási feladatokat megoldó branch-and-bound módszerektől, mivel a szétválasztó lépés a kombinatorikusan lehetséges struktúrákat figyelembe véve történik.

A részproblémákat a döntés leképezés segítségével definiáljuk. Egy részproblémához tartozó döntés leképezést a branch-and-bound algoritmus

fájának gyökerétől a részproblémát reprezentáló csúcsig vezető lépések során hozott döntések halmaza adja meg. Az újabb szétválasztó lépés egy újabb döntés meghozatalát jelenti, amely a (i) részproblémához tartozó kombinatorikusan lehetséges megoldások particionálását jelenti, (ii) egy újabb konzisztens döntés leképezést eredményez, amely a szülő részproblémához tartozó döntés leképezés kiterjesztése.

A folyamat-hálózatszintézis feladat megoldásakor a P_i - $\delta_i[m_i]$ konzisztens döntés leképezéssel definiált - részproblémát $S(\delta_i[m_i])$-vel jelöljük és azokat a megoldásokat tartalmazza, amelyekhez tartozó P-gráfnak az aktuális részprobléma döntés leképezése által definiált P-gráf részgráfja és nem tartalmaznak az eddigi döntésekkel kizárt műveleti egységeket; azaz $S(\delta_i[m_i]) = \{\text{gráf}(\delta_k[m_k]) \mid \delta_k[m_k] \geq \delta_i[m_i]\}$.

A branch-and-bound fa gyökeréhez a $\varnothing$ döntés leképezés, és ennek megfelelően az összes kombinatorikusan lehetséges struktúra tartozik, míg a branch-and-bound fa leveleihez tartozó döntés leképezések egy kombinatorikusan lehetséges megoldás P-gráfját adják meg.

A szétválasztó függvénynek az aktuális részproblémához tartozó döntés leképezés alapján kell új döntés leképezéseket megadnia úgy, hogy azok az aktuális részproblémához tartozó döntés leképezés kiterjesztései legyenek, és a hozzájuk tartozó részproblémák megfeleljenek a szétválasztó függvény általános definíciójában leírtaknak. A keresés hatékonyságát növeli, hogy a "diszjunkt leszármazottak" tulajdonság is teljesül.

Ezek alapján a szétválasztó lépés: keressünk egy olyan anyagot, amely minden $S(\delta_i[m_i])$-beli struktúra része, de még nem volt döntés az előállításáról. Ezen anyagok halmazát az alábbi p leképezés segítségével határozhatjuk meg:

$$p(S(\delta_i[m_i])) = (\text{mat}^{\text{in}}(\text{op}(\delta_i[m_i])) \cup P) \setminus (m_i \cup R), \text{ ahol } \text{mat}^{\text{in}}(o) = \{x \in \text{in}_u \mid u \in o\}.$$

Ha ilyen anyag nincs, az $S(\delta_i[m_i])$ részproblémában a megoldások struktúrája teljesen definiált, azaz egyértelmű, így a korlátozó függvény tulajdonságának megfelelően ennél a részproblémánál nincs szükség további szétválasztásra, a részprobléma levél. Ha a $p(S(\delta_i[m_i]))$ halmaz nem üres, valamely x elemének kiválasztása után az összes lehetséges módon, a konzisztencia megtartásával,

bővítjük az $S(\delta_i[m_i])$ részprobléma döntés leképezését, így megkapjuk a részprobléma gyerekeit definiáló döntés leképezéseket. Előfordulhat, hogy bizonyos anyagok esetén csak egy lehetséges döntést hozhatunk, így nem történik valódi szétválasztás. A maximális neutrális kiterjesztés alkalmazásával [16] ez az eset elkerülhető: minden döntés után (valamint a kezdő lépésben) megvizsgáljuk, hogy szétválasztó lépés alkalmazása nélkül tovább bővíthető-e az aktuális döntés leképezés, majd ezt követi a korlátozó függvény hívása és az esetleges újabb szétválasztás. A maximális neutrális kiterjesztést alkalmazva garantált, hogy az újabb döntések minden esetben valódi szétválasztást eredményeznek. Az ABB algoritmus szétválasztó függvényének pontos definiálásához meg kell adnunk egy "anyagválasztó függvényt" ($x = A(S(\delta_i[m_i])) \in p(S(\delta_i[m_i]))$, ami lehet a legkisebb indexű anyag (a feladat definíció alapján) vagy a legkevesebb módon előállítható anyag, de ez lényegében tetszőleges, az ABB algoritmus működését nem befolyásolja lényegesen.

Az ABB algoritmus szétválasztó függvénye valamely $S(\delta_i[m_i])\neq\varnothing$ részproblémára:

$son(S(\delta_i[m_i])) = \{S(\delta_{ij}[m_{ij}]) \mid \delta_k[m_k] = \{(\delta_i[m_i] \cup \{(x, c)\}, x = A(S(\delta_i[m_i])), c \in \wp(\Delta(x)) \setminus \{\varnothing\}, \delta_k[m_k]$ konzisztens, δ_{ij} a δ_k maximális neutrális kiterjesztése$\}$

Ez a függvény megfelel mint szétválasztófüggvény [16].

A feldolgozásra váró részproblémák közül az aktuális részproblémát kiválasztó keresőfüggvény lehet például a hagyományos branch-and-bound algoritmusoknál is alkalmazott depth-first stratégiának megfelelően választó függvény.

Ezzel definiáltuk a folyamat-hálózatszintézis feladat megoldására kidolgozott gyorsított branch-and-bound szétválasztó lépését, amely a feladat matematikai modelljétől függetlenül, minden folyamat-hálózatszintézis feladat megoldására alkalmazható.

A döntés leképezés alapján egy adott részproblémánál polinomiális időben meghatározhatjuk a részproblémához tartozó összes megoldásstruktúrában szereplő műveleti egységeket, a kizárt műveleti egységeket és a választható műveleti egységeket. Ezek segítségével a műveleti egységek modelljét és ha

szükséges valamilyen relaxáló módszert felhasználva, már meghatározható a korlátozó függvény.

10. A folyamat-hálózatszintézis feladat kiterjesztései

Az előző fejezetben ismertetett ABB algoritmus alkalmas alapvető változtatás nélkül több feladatosztály megoldására is, sok feladattípus megoldható a korlátozó eljárás módosításával a megoldás keresését vezérlő algoritmus kombinatorikus lépéseinek módosítása nélkül. Az ABB algoritmus ezeken kívül sok olyan feladatosztály megoldásának alapjául is szolgálhat, amelyek hatékony megoldása az algoritmus lényegi módosítása nélkül nem lehetséges.

Ebben a fejezetben az alap folyamat-hálózatszintézis feladat kiterjesztéseivel foglalkozunk, a kiterjesztések mint feladatosztályok megoldására szolgáló módszereket mutatunk be, amelyek mindegyike az ABB algoritmusra épül. Ezekben a feladatosztályokban már a megoldásstruktúra definíciója is módosulhat, ennek megfelelően változnak - általában bővülnek - a megoldásstruktúrák tulajdonságait leíró axiómák és így az ABB algoritmus lényegi, vezérlő része is. A fejezetben bemutatott kiterjesztések közül az első kettő is ilyen típusú. Az első az alapfeladatban definiált kötelezően előállítandó termékek mellett a megoldásstruktúra összes lehetséges outputját megadja, míg a második kiterjesztésben már a tervezés során bizonyos egyszerűsített irányíthatósági megkötéseket veszünk figyelembe.

A harmadik kiterjesztés eltér az előzőektől, a rendszert alkotó műveleti egységek matematikai modellje változik lényegesen, a megoldásstruktúrák definíciója változatlan marad. Ennek a feladatosztálynak a megoldása lehetséges az ABB algoritmus vezérlő részének változtatása nélkül, ugyanakkor ebben az esetben a korlátozó eljárás a műveleti egységek szakaszosan folytonos, nem konkáv költségfüggvénye miatt önmagában is nehéz feladat, lényegében egy újabb branch-and-bound algoritmus szükséges a megoldásához. Az ismertetett módszer ezt a második branch-and-bound algoritmust integrálja az ABB algoritmusba, lehetővé téve a feladat kombinatorikus jellegének jobb kihasználását.

A negyedik kiterjesztés lényegében a megoldó módszer technikai jellegű kiterjesztését jelenti, itt az ABB algoritmus párhuzamos működési elvű számítógépeken futtatható változatát mutatjuk be.

Mind az itt ismertetett, mind további kiterjesztések [13, 19, 22] – például többlépcsős (multiperiódikus) folyamat-hálózatszintézis, hibatűrő irányítórendszerrel integrált folyamat-hálózatszintézis – kombinált alkalmazása lehetséges. Így az ABB algoritmusra alapozva akár olyan módszer is megadható, amely többlépcsős, hulladékkezeléssel integrált folyamat-hálózatszintézis feladatot old meg szakaszosan folytonos költségfüggvénnyel adott műveleti egységek esetén. Természetesen ez a módszer is megvalósítható párhuzamos feldolgozású számítógépeken.

10.1. Hulladékkezeléssel integrált folyamat-hálózatszintézis

Mint azt már korábban definiáltuk, a folyamat-hálózatszintézis alapfeladata a rendelkezésre álló műveleti egységek egy részhalmazának és azok működési paramétereinek megkeresése, amelyek azt a hálózatot alkotják, amely a kívánt termékeket minimális költséggel állítja elő. Ez már önmagában is nehéz feladat, részben a kombinatorikus jellege, részben a műveleti egységek esetlegesen bonyolult matematikai modellje miatt.

A hulladékkezeléssel integrált folyamat-hálózatszintézis feladat megoldása az alapfeladatnál is bonyolultabb, hiszen kombinatorikus szempontból a termék gyártásában részt vevő műveleti egységek mellett egyidejűleg a hulladékkezelő rendszer műveleti egységeit is ki kell választani, matematikai programozási szempontból a részfeladatok mérete és száma is nagyobb.

Mint az irodalmi áttekintésben is bemutattuk, az eddig kidolgozott módszerek ennél az összetett feladatnál nem integrálták teljesen a termékgyártást és a hulladékkezelést, azaz két kisebb folyamat-hálózatszintézis feladatként oldották meg a feladatot. Mivel a két rész nem független, ez a megközelítés könnyen eredményezhet az optimális megoldásnál lényegesen rosszabb megoldást, egy olcsóbb termelő hálózathoz tervezett hulladékkezelő rendszer lényegesen drágábbá teheti a komplett megoldást, mintha a termelő rész szempontjából egy

költségesebb megoldást választottunk volna. Nyilvánvaló, hogy optimális megoldást csak az együttes tervezés adhat.

A hulladékkezeléssel integrált folyamat-hálózatszintézis feladat megoldására kidolgoztuk az ABB algoritmus módosított változatát [15, 39].

Az alap folyamat-hálózatszintézis feladathoz hasonlóan a műveleti egységek modellezésével itt sem foglalkozunk, bár fontos feladat, továbbra is feltételezzük, hogy megfelelő technikák erre rendelkezésre állnak és a kész hálózatok optimális működését is meghatározó költségfüggvény-számítás (valamely matematikai programozási módszer) is adott.

A rendelkezésre álló (már megépített) műveleti egységek költségfüggvényének megfelelő módosításával a kidolgozott módszer alkalmas már meglevő rendszerek hulladékkezeléssel integrált újratervezésére, ha szükséges a termelőrendszert is módosítva.

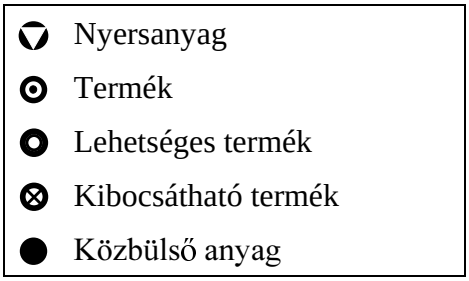
10.1.1. Feladat definíció

A hulladékkezeléssel integrált folyamat-hálózatszintézis a folyamat-hálózatszintézis alapfeladatát a teljes output definícióval bővíti. A szükséges termékek megadásán túl a további output anyagoknak is teljesíteniük kell további feltételeket.

Bizonyos anyagok nem lehetnek megoldás outputjai, ezeket közbülső anyagoknak nevezzük. Az általánosságra törekedve a termékek mellett megengedett outputokat tovább osztályozzuk kibocsátható anyagokra és lehetséges termékekre. Így a struktúrákban előforduló anyagokat 5 csoportba sorolhatjuk. A **nyersanyagok** és a kötelezően gyártandó **termékek** definíciója azonos az alapfeladat definíciójában leírtakkal. **Közbülső anyag** nem lehet megoldás outputja. A **lehetséges termékek** és a **kibocsátható anyagok** lehetnek a rendszer outputjai, amíg az előbbi gyártása esetén értékesíthető terméknek minősül, azaz csökkentheti a megoldás költségét, addig a kibocsátható anyagok gyártása a konkrét feladat célfüggvényétől függően akár növelheti is a költségeket: ezen anyagok gyártása megengedett, de büntetőköltséget eredményezhet. A hulladékkezeléssel integrált folyamat-hálózatszintézis feladata a műveleti egységek egy olyan részhalmazának

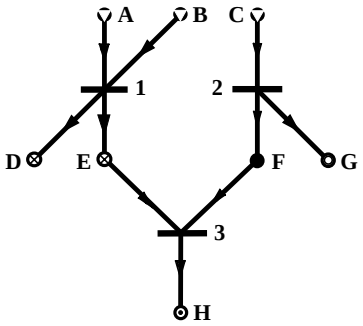
megkeresése a megfelelő állapotváltozókkal, amelyekből felépülő hálózat (megoldás) az adott inputok (nyersanyagok) segítségével a kért outputokat (termékek) minimális költséggel állítja elő, továbbá a hálózatnak nem outputja "közbülső anyag" típusú anyag. Ez a feladat definiálható a termékek, a lehetséges termékek, a nyersanyagok, a közbülső anyagok és a műveleti egységek megadásával.

A megoldások struktúráinak reprezentációjára az alapesethez hasonlóan a P-gráfot használjuk, a különböző típusú anyagok megkülönböztetésére használt szimbólumokat a 2. ábra mutatja.



2. ábra. Az öt anyagcsoport szimbólumai a P-gráf reprezentációban.

Illusztrációként a 3. ábra az A, B, C nyersanyagokból H terméket gyártó struktúra P-gráfját mutatja be, amelynek a termék mellett a G lehetséges termék és a D kibocsátható anyag az outputja.



3. ábra. Az ({A,B,C,D,E,F,G,H}, {1,2,3}) P-gráf.

10.1.2. Kombinatorikusan lehetséges struktúrák

A módosított feladat definíció miatt nyilván változik a megoldások halmaza is. A megoldás hálózatokban bizonyos műveleti egységek már nem direkt vagy indirekt módon a termék gyártását szolgálják, hanem az integrált hulladékkezelő

rendszer részei. Ennek megfelelően a megoldások struktúrái is módosulnak, azaz a kombinatorikusan lehetséges megoldások halmaza is eltér, így a tulajdonságaikat definiáló axiómákat is módosítanunk kell. A hulladékkezeléssel integrált folyamat-hálózatszintézis feladat megoldásstruktúráit leíró módosított axiómák az alábbiak:

(SW1) Minden termék része a struktúrának.

(SW2) Egy a struktúrában szereplő anyag akkor és csak akkor nyersanyag, ha egyetlen a struktúrában szereplő műveleti egység sem állítja elő.

(SW3) Minden a struktúrában szereplő műveleti egység a folyamat-hálózatszintézis feladatban definiált.

(SW4) Minden a struktúrában szereplő y műveleti egységre létezik utak sorozata, ahol az y műveleti egységet reprezentáló csúcs az első út eleje, az utolsó út vége valamely (lehetséges) terméket előállító műveleti egységet reprezentáló csúcs; mindegyik út első és utolsó eleme műveleti egységet reprezentáló csúcs; mindegyik útnak közös az eleje vagy a vége a következő úttal és az utakban nincs közös anyagot reprezentáló csúcs.

(SW5) Ha egy anyag része a struktúrának, akkor létezik a struktúrában olyan műveleti egység, amelynek az inputja vagy outputja.

(SW6) Ha valamely anyagot nem dolgoz fel egyetlen műveleti egység sem, akkor az nem közbülső anyag típusú.

Mint látható az (SW1), (SW2), (SW3) és (SW5) axiómák azonosak az (S1), (S2), (S3) és (S5) axiómákkal, az általuk definiált tulajdonságok erre a feladatosztályra is érvényesek. Az alapesetben az (S4) axióma biztosította, hogy csak termék gyártásában résztvevő műveleti egységek szerepeljenek a vizsgált struktúrákban. A hulladékkezeléssel való integrálás miatt itt a "termelő" műveleti egységek mellett "hulladékkezelő" műveleti egységeket is meg kell engednünk. Az (SW4) axiómát a 2. példa szemlélteti.

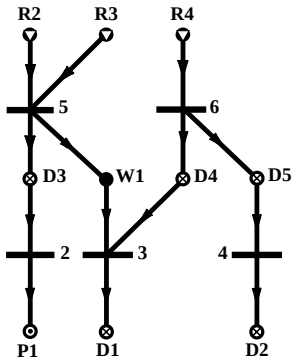
Az újonnan bevezetett (SW6) axióma a közbülső anyagként definiált anyagokat kibocsátó struktúrákat zárja ki a lehetséges megoldásstruktúrák közül.

2. példa. Tekintsük a P1 terméket az R1, R2, R3, R4 nyersanyagokból az 1. táblázatban adott műveleti egységek segítségével előállító hulladékszintézissel integrált folyamat-hálózatszintézis feladatot. Mivel a példát a strukturális tulajdonságok szemléltetésére használjuk, a műveleti egységeket csak az input-output halmazaikkal adjuk meg, pontosabb modellezésükkel nem foglalkozunk. A fent említett anyagokon kívül a W1 közbülső anyag, a többi, azaz D1, D2, D3, D4 és D5 kibocsátható anyagok.

1. táblázat. A 2. példa műveleti egységei.

#	Input	Output
1	R1, D3	P1
2	D3	P1
3	W1, D4	D1
4	D5	D2
5	R2, R3	D3, W1
6	R4	D4, D5

Tekintsünk egy a fenti műveleti egységekből épített struktúrát, amely a 4. ábrán látható. Az SW1, SW2, SW3, SW5, és SW6 axiómáknak nyilván megfelel a struktúra. Vizsgáljuk meg, hogy a struktúrában szereplő műveleti egységek megfelelnek-e az SW4 axióma feltételeinek. A 2. táblázat a műveleti egységekhez tartozó, az SW4 axióma feltételeinek megfelelő útsorozatokat tartalmazza.



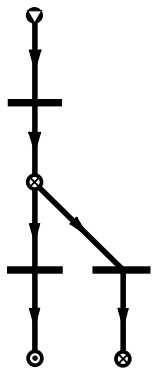
4. ábra. A 2. példa műveleti egységeiből álló P-gráf.

2. táblázat. A 2. példa műveleti egységeihez tartozó útsorozatok.

műveleti egység	útsorozat
2	(2)
3	(5,3), (5,2)
4	(4),(6,4), (6,3), (5,3), (5,2)
5	(5,2)
6	(6,3), (5,3), (5,2)

Mivel a többi axiómát is teljesíti a 4. ábrán látható struktúra, ezért kombinatorikusan lehetséges struktúra. A struktúrában a 2. és az 5. műveleti egység szerepe a termék gyártása, a 3. műveleti egység már a struktúra "hulladékkezelő" részéhez tartozik, a W1 közbülső anyagot dolgozza fel. A 6. műveleti egység a hulladékfeldolgozáshoz szükséges D4 anyagot állítja elő, míg a 4. műveleti egység a D5 anyagot dolgozza fel. Ez a műveleti egység egy kibocsátható anyagot dolgoz fel, azaz elhagyható lenne, viszont a D5 anyag helyett a D2 kibocsátását indokolttá teheti a költségfüggvény.

Az alapesethez hasonlóan itt is azon struktúrákat nevezzük kombinatorikusan lehetségesnek, melyek P-gráfjai megfelelnek az axiómáknak. Az 5. ábra egy kombinatorikusan nem lehetséges struktúrát mutat be (azaz a struktúra nem lehet optimális megoldás struktúrája).



5. ábra. Az SW4 axiómát nem teljesítő struktúra P-gráfja.

10.1.3. MSGW algoritmus

Az SW1, SW2, ..., SW6 axiómák drasztikusan csökkentik azon műveleti egység-kombinációk számát, amelyeket lehetséges megoldásként figyelembe kell vennünk a hulladékkezeléssel integrált folyamat-hálózatszintézis során. Az

alap folyamat-hálózatszintézis feladathoz hasonlóan az itt vizsgált feladatosztály megoldásai is zártak az unió műveletre.

1. tétel. A hulladékkezeléssel integrált folyamat-hálózatszintézis feladat megoldásstruktúrái zártak az únió műveletre.

Bizonyítás: Elegendő megmutatni, hogy két megoldásstruktúra P-gráfjának (jelölje ezeket P1, illetve P2) uniójaként kapott P-gráf is teljesíti az SW1-SW6 axiómákat, azaz szintén kombinatorikusan lehetséges struktúrát reprezentál.

SW1: A P1∪P2 gráf tartalmazza mind a P1, mind a P2 gráf csúcsait, tehát tartalmazza a termékeket is.

SW2: Sem a P1, sem a P2 gráf nem tartalmaz nyersanyag típusú csúcsba vezető élet, így ez teljesül a P1∪P2 gráfra is.

SW3: Nyilvánvalóan teljesül.

SW4: Ha egy műveleti egység típusú csúcs része a P1∪P2 gráfnak, akkor eleme a P1 vagy a P2 gráfnak. De ekkor az SW4 axióma feltételeit teljesítő P1 vagy P2 gráfbeli útsorozat is része a P1∪P2 gráfnak, azaz a P1∪P2 gráf is tejesíti az axióma feltételeit.

SW5: Nyilvánvalóan teljesül.

SW6: Mivel sem a P1, sem a P2 gráf nem tartalmaz olyan közbülső anyag típusú csúcsot, amelyből nincs kivezető él műveleti egység típusú csúcshoz, ezért ilyen csúcs a P1∪P2 gráfban sincs.

Az alapesettel analóg módon tehát létezik maximális struktúra, amely itt is megadja a minimális keresési teret az optimális megoldás kereséséhez. A maximális struktúra generálására kidolgoztuk a polinomiális idejű MSGW algoritmust, amely az alap folyamat-hálózatszintézis feladatra kidolgozott MSG algoritmus kiterjesztése. A 6. ábra az MSGW algoritmus Pidgin Algol nyelvű változatát mutatja be.

10.1.4. Az algoritmus helyességének bizonyítása

Az algoritmus két fő részből áll; az első részben (az st4 utasításig) azon műveleti egységek kizárása történik, amelyek biztosan nem lehetnek

Input: O*: a műveleti egységek halmaza* M*: az anyagok halmaza*
I*: a közbülső anyagok halmaza* P_r*: a termékek halmaza*
P_p*: a lehetséges termékek halmaza* R*: a nyersanyagok halmaza*

Jelölések: $o \subseteq O$, $mat^{in}(o)=\{x \in M \mid u \in o, x \in in_u\}$, $mat^{out}(o)=\{x \in M \mid u \in o, x \in out_u\}$
$m \subseteq M$, $op^{in}(m)=\{u \in O \mid m \in out_u\}$, $op^{out}(o)=\{u \in O \mid m \in in_u\}$

Azaz az op^{in}, mat^{in} *függvények egy csúcshalmazba bejövő élek kezdőpontjainak halmazát, míg az* op^{out}, mat^{out} *függvények egy csúcshalmazból kiinduló élek végpontjainak halmazát határozzák meg.*

procedure msg_w():
begin
st1: $O := O \setminus op^{in}(R)$;
rp1: **repeat**
st2: $M := mat^{in}(O) \cup mat^{out}(O)$; $r := mat^{in}(O) \setminus (mat^{out}(O) \cup R)$;
wh1: **while** r is not empty **do**
begin
$x \in r$; $M := M \setminus \{x\}$; $o := op^{out}(x)$; $O := O \setminus o$;
$r := (r \cup (mat^{out}(o) \setminus mat^{out}(O))) \setminus \{x\}$;
end;
co1: **if** $P_r \cap M \neq P_r$ **then stop**; **comment**: there is no maximal structure
st3: $P_p := P_p \cap M$; $r := (I \cap M) \setminus mat^{in}(O)$; $o_w := \varnothing$;
wh2: **while** r is not empty **do**
begin
$x \in r$; $M := M \setminus \{x\}$; $o := op^{in}(x)$; $O := O \setminus o$;
$r := (r \cup ((mat^{in}(o) \setminus mat^{in}(O)) \cap I)) \setminus \{x\}$; $o_w := o_w \cup o$;
end;
until $o_w = \varnothing$;
st4: $p := (P_r \cup P_p)$; $O_u := \varnothing$; $O_d := \varnothing$; $m_u := \varnothing$; $m_d := \varnothing$; $r := \varnothing$;
rp2: **repeat**
wh3: **while** p is not empty **do**
begin
$x \in p$; $m_u := m_u \cup \{x\}$; $o_x := op^{in}(x) \setminus O_u$;
$r := r \cup (mat^{out}(o_x \setminus O_d) \setminus (\{x\} \cup m_d))$;
$O_u := O_u \cup o_x$; $p := (p \cup mat^{in}(o_x)) \setminus (R \cup m_u)$;
end;
wh4: **while** r is not empty **do**
begin
$x \in r$; $m_d := m_d \cup \{x\}$;
$o_x := op^{out}(x) \setminus O_d$;
$p := p \cup (mat^{in}(o_x \setminus O_u) \setminus (\{x\} \cup R \cup m_u))$;
$O_d := O_d \cup o_x$; $r := (r \cup mat^{out}(o_x)) \setminus m_d$;
end;
until p is empty;
st5: $O_r := O_u \cup O_d$;
st6: $M_r := mat^{in}(O_r) \cup mat^{out}(O_r)$; **write**(M_r, O_r);
end;

6. ábra. Az MSGW algoritmus.

megoldásstruktúra részei, a második rész a maximális struktúra felépítése. A wh1 ciklus a nem előállítható, nem nyersanyag inputot használó műveleti egységeket törli, ez a ciklus az alap folyamat-hálózatszintézis feladathoz kidolgozott MSG algoritmusnak is része. A wh2 ciklus a nem felhasználható közbülső anyagokat előállító műveleti egységeket zárja ki a további vizsgálatból. Mivel ez a két ciklus nem független (mindegyik ugyanabból az O műveleti egység halmazból töröl elemeket) - például egy műveleti egységet a wh2 ciklusban törölve ismét lehetséges olyan műveleti egység, amelynek valamely nem nyersanyag inputja nem előállítható -, ezért addig ismételjük őket (rp1), amíg egyik ciklus sem töröl újabb műveleti egységet.

A wh3 ciklus először a termékeket előállító műveleti egységeket határozza meg és adja hozzá a maximális struktúra műveleti egységeinek halmazához, a későbbiekben a wh4 ciklust követve (az rp2 ismételt futásakor) a maximális struktúrába (a wh4 ciklusban) felvett (hulladékkezelő) műveleti egységek inputjait előállító műveleti egységeket határozza meg. A wh4 ciklus a wh3 által kiválasztott műveleti egységek outputjait feldolgozó (hulladékkezelő) műveleti egységeket adja meg.

Mivel a műveleti egységek halmaza véges, valahány lépés után újabb műveleti egység nem választható ki. A közbülső anyagokat felhasználó, illetve a nem nyersanyag inputot gyártó műveleti egységek létezését az rp1 ciklus garantálja.

2. tétel. Az MSGW algoritmus véges, és polinomiális időben véget ér.

Bizonyítás: Mivel az algoritmus nem hív meg külső algoritmusokat és nem rekurzív, elegendő belátni, hogy a ciklusok végesek és minden ciklus lehetséges ismétléseinek száma polinomiális a műveleti egységek számát tekintve.

A wh1 és wh2 ciklusokban a véges O műveleti egység halmazból elemeket törlünk, ezért a két ciklus és így az rp1 ciklus végrehajtása is csak véges sokszor ismétlődhet (ha az O üres az op^{in} illetve op^{out} halmazok is üresek).

A wh3 ciklus minden ismétlésekor az O_u halmaz bővül, míg a wh4 ciklusban az O_d halmaz. Mivel a lehetséges műveleti egységek száma véges, az algoritmus első részéhez hasonlóan a wh3, wh4, rp2 ciklusok is csak véges sokszor ismétlődnek, az algoritmusban a műveletek száma polinomiális a műveleti egységek számát tekintve.

3. tétel. Az rp2 ciklus i. futásakor az O_u halmazba azon műveleti egységek kerülnek, amelyekből létezik 2i-1 útból álló - az SW4 axióma feltételeinek megfelelő - útsorozat. Hasonlóan az O_d halmazba azon műveleti egységek kerülnek, amelyekből létezik 2i útból álló - az SW4 axióma feltételeinek megfelelő - útsorozat.

Bizonyítás: Az rp2 ciklus 1. lefutása után (i=1):

(i) Az O_u halmaz azon műveleti egységeket tartalmazza, amelyekre létezik egy (lehetséges) termékhez vezető út: nyilvánvaló, a wh3 ciklus az r halmazt módosító utasítást elhagyva azonos az MSG algoritmus [12] "építő" részével.

(ii) Az O_d halmaz elemeiből két - az SW4 axióma feltételeinek megfelelő - utat megadva eljuthatunk egy (lehetséges) termékhez: az r halmaz a wh4 ciklus indulásakor azokat az x anyagokat tartalmazza, amelyeket valamely $y \in O_u$ műveleti egység állít elő, és létezik olyan út y-ból (lehetséges) termékhez, amelynek x nem része. Mivel a wh4 ciklusban O_d elemei az r elemeiből kiinduló utak műveleti egységei lesznek, a fenti (ii) állítás teljesül, a tétel i=1-re igaz.

Tegyük fel, hogy az állítás igaz i-1-re.

Ekkor az rp2 ciklus i-1. végrehajtása után a p halmaz elemei azon x anyagok lesznek, amelyek egy olyan $y \in O_d$ műveleti egység inputjai, amelyhez létezik egy 2i-2 útból álló (lehetséges) termékhez vezető útsorozat (p_1, p_2, ..., p_{2i-2}) és x ezek egyikének sem eleme. Így a wh3 ciklus következő futásakor pontosan azok a z műveleti egységek kerülnek az O_u halmazba, amelyekre a [z, ..., x, y], p_1, p_2, ..., p_{2i-2} útsorozat megfelel az SW4 axióma feltételeinek. Az i=1 esethez hasonlóan belátható a wh4 ciklusban felvett műveleti egységekre az eggyel hosszabb útsorozat létezése.

4. tétel. Az MSGW algoritmus által meghatározott (M_r, O_r) struktúra kombinatorikusan lehetséges.

Bizonyítás: Megmutatjuk, hogy az (M_r, O_r) struktúra teljesíti az axiómákat.

SW1: Mivel a wh3 ciklus kezdetén minden termék eleme a p halmaznak, a ciklus befejezésekor az O_u halmaz műveleti egységei gyártják az összes terméket, így az st6 utasítás után minden termék eleme lesz az M_r halmaznak.

SW2: A nyersanyagokat gyártó műveleti egységeket az st1 utasítás kizárja a vizsgálatból. A wh1 ciklus indulásakor az r halmaz azon nem nyersanyagok halmaza, amelyet az aktuális O műveleti egység halmaz elemei felhasználnak, de nem állítanak elő. Ez a tulajdonság a ciklus futása során igaz (ciklus invariáns), a ciklus célja az ilyen anyagokat felhasználó műveleti egységek kiszűrése a maximális struktúra lehetséges műveleti egységei közül. A wh1 ciklus befejeződésekor tehát nincs ilyen anyag. Az rp1 ciklus akkor fejeződik be, amikor a wh1 ciklus után a wh2 ciklus nem töröl újabb műveleti egységeket, így ez az rp2 ciklus indulásakor is igaz, azaz minden műveleti egység minden inputja vagy nyersanyag, vagy előállítható más műveleti egységekkel. Így ez a wh3 ciklusban minden O_u, illetve a wh4 ciklusban minden O_d halmazba felvett műveleti egység minden inputjára is igaz, azaz O_r minden elemére is.

SW3: Nyilvánvalóan teljesül.

SW4: A 3. tételből következik.

SW5: Nyilvánvalóan teljesül.

SW6: A wh4 ciklus a műveleti egységek outputjait feldolgozó műveleti egységeket felveszi a maximális struktúrába, a wh2 ciklus garantálja, hogy ez közbülső anyag típusú outputra mindig lehetséges.

5. tétel. Az MSGW algoritmus által generált struktúra a feladat maximális struktúrája.

Bizonyítás: Tegyük fel, hogy létezik olyan u műveleti egység, amely nem eleme O_r-nak, de eleme valamely kombinatorikusan lehetséges megoldásstruktúrának.

Ha egy műveleti egységet a maximális CFC struktúra lehetséges műveleti egységei közül az st1 utasítás zár ki, az nyersanyagként definiált anyagot állít elő, így az (SW2) axióma szerint nem lehet megoldás része. A további műveleti egységeket kizáró utasítások a ciklusokon belül vannak, ezért mindig azt kell megvizsgálnunk, hogy az O_r illetve O műveleti egység halmaz aktuális műveleti egységeit tekintve indokolt-e az u műveleti egység kizárása a további vizsgálatból.

Ha az rp1 ciklusban töröltük az u műveleti egységet, akkor vagy az O műveleti egység halmaz elemeivel elő nem állítható nem nyersanyag inputja van (törlés a

wh1 ciklusban), vagy az O műveleti egység halmaz elemeivel nem feldolgozható közbülső anyag típusú outputja van (törlés a wh2 ciklusban), tehát nem lehet része az O műveleti egység halmaz elemeiből képzett megoldásstruktúrának.

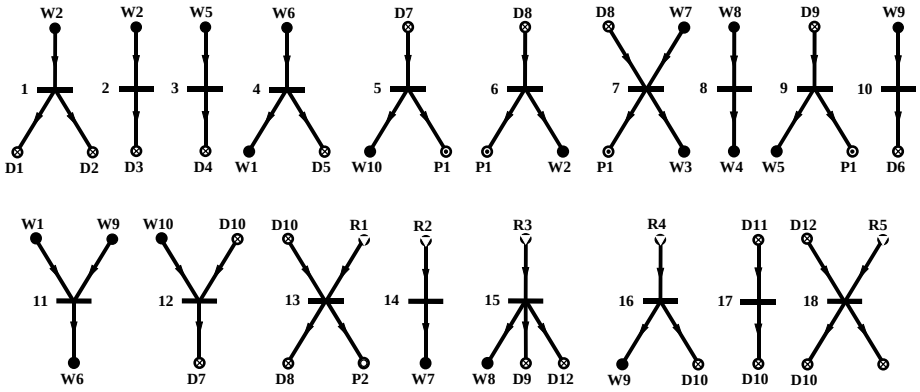
Ha az u része valamely kombinatorikusan lehetséges struktúrának, akkor létezik egy az SW4 axióma feltételeinek megfelelő n hosszú útsorozat, ekkor azonban az rp2 ciklus n/2-ik végrehajtásakor u-nak be kellett kerülnie az O_u vagy az O_d műveleti egység halmazba, ami ellentmondásra vezet.

6. tétel. Az MSGW algoritmus akkor és csak akkor nem ad struktúrát eredményként, ha feladathoz nem létezik maximális struktúra.

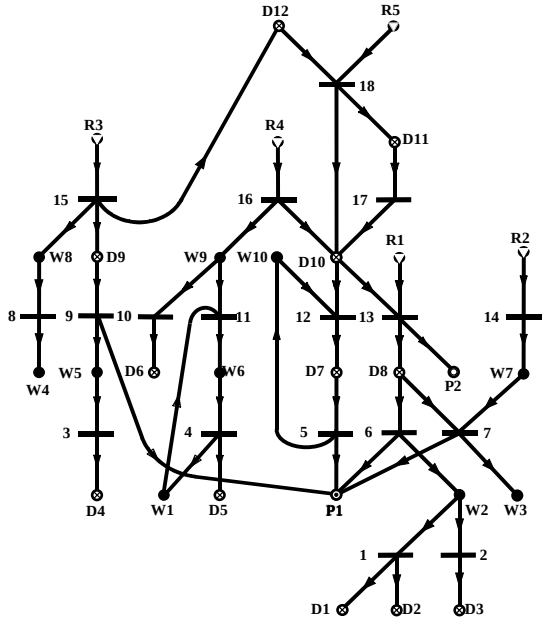
Bizonyítás: Az algoritmus szerint nincs maximális struktúra, ha az algoritmus végrehajtása során egy O műveleti egység halmazra a co1 feltétel teljesül. Ekkor az O műveleti egység halmaz elemeiből nyilván nem adható meg megoldásstruktúra. A 5. tétel bizonyításában megmutattuk, hogy a törölt műveleti egységek nem lehetnek egyetlen megoldásstruktúra részei sem, így az inputként adott műveleti egység halmaz elemeiből sem adható meg megoldásstruktúra, így a feladatnak nem létezik megoldása, és ezzel együtt maximális struktúrája sem.

A 5. tételből következik, hogy ha az algoritmus ad struktúrát eredményként, akkor az a maximális struktúra.

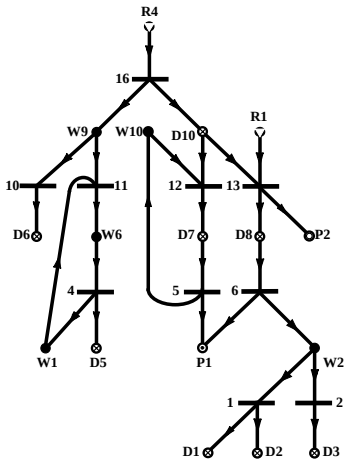
3. példa. Tegyük, fel hogy 7. ábrában adott műveleti egységek segítségével a P1 terméket kell gyártanunk. A többi anyag osztályozása leolvasható az ábráról. A feladathoz tartozó P-gráfot a 8. ábra tartalmazza. A 9. ábra a maximális struktúrát mutatja be. Összehasonlításképp a 10. ábrában megadjuk azt a maximális struktúrát, amelyet szintén a 7. ábra műveleti egységeiből kiindulva kaphatunk, de az integrált hulladékkezelés elhagyásával. A 11. ábra néhány kombinatorikusan lehetséges megoldást illusztrál.



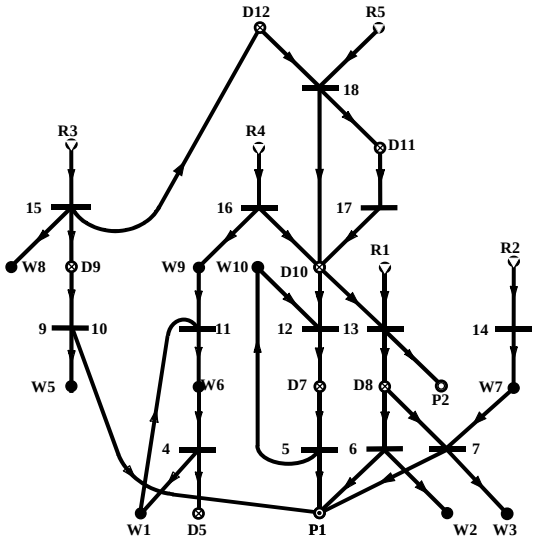
7. ábra. A 3. példa műveleti egységei.



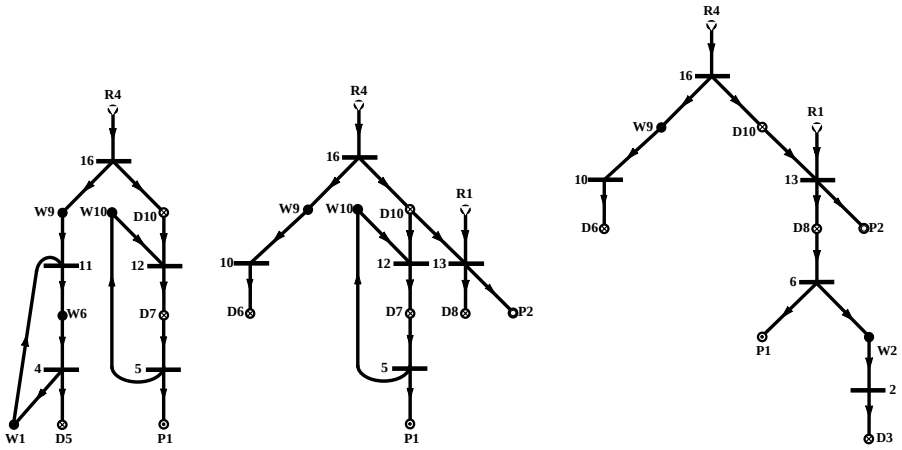
8. ábra. A 3. példa P-gráfja.



9. ábra. A 3. példa maximális struktúrája.



10. ábra. A 3. példa maximális struktúrája a hulladékkezelés figyelembevétele nélkül.



11. ábra. A 3. példa néhány kombinatorikusan lehetséges struktúrája.

Mint látható, néhány a feladat definíciójában szereplő műveleti egység nem része a maximális struktúrának, ezeket a műveleti egységeket már nem kell figyelembe vennünk az optimális megoldás keresése során.

10.1.5. ABBW algoritmus

Az ABB algoritmushoz hasonló módon a hulladékkezeléssel integrált folyamat-hálózatszintézis feladatot megoldó ABBW algoritmusra is definiáljuk a megfelelő függvényeket. A részproblémák definíciója az alapesettel azonos, a szétválasztó lépés során is a megoldások struktúráinak fokozatos felépítésével kereshetjük az optimális megoldást. Mivel ebben az esetben is a termék(ek)et minden megoldásstruktúrának tartalmaznia kell, ezeket tekinthetjük

kiindulópontnak. Az alapesettől eltérően azonban itt választási lehetőség nem csak egy anyag gyártásánál adódik, hanem az integrált hulladékkezelés miatt bizonyos esetekben egy anyag további feldolgozásáról is döntenünk kell. Ennek megfelelően a szétválasztó függvényt módosítanunk kell és a szétválasztó függvény alapjául szolgáló döntés leképezést is ki kell terjesztenünk. A kiterjesztett döntés leképezés esetén használt definíciók, illetve jelölések az alábbiak:

Jelölje Δ_{in} azt a leképezést az anyagok és a műveleti egységek hatványhalmaza között, amely minden $X \in M$ anyagra megadja az X-et előállító műveleti egységek halmazát, azaz $\Delta_{in}(X) = \{i \in O \mid X \in out_i\}$. Továbbá jelölje Δ_{out} azt a leképezést az anyagok és a műveleti egységek hatványhalmaza között, amely minden $X \in M$ anyagra megadja az X-et felhasználó műveleti egységek halmazát, azaz $\Delta_{out}(X) = \{i \in O \mid X \in in_i\}$.

10. definíció. Legyen $m \subseteq M$ és $\delta_{in}(X) \subseteq \Delta_{in}(X)$ minden $X \in m$-re. Ekkor δ_{in}, mint leképezés az m halmazból a műveleti egységek részhalmazainak halmazába, $\delta_{in}[m] = \{(X, \delta_{in}(X)) \mid X \in m\}$, a **gyártó döntés leképezés** m felett.

11. definíció. Legyen $m \subseteq M$ és $\delta_{out}(X) \subseteq \Delta_{out}(X)$ minden $X \in m$-re. Ekkor δ_{out}, mint leképezés az m halmazból a műveleti egységek részhalmazainak halmazába, $\delta_{out}[m] = \{(X, \delta_{out}(X)) \mid X \in m\}$, a **felhasználó döntés leképezés** m felett.

12. definíció. Legyen $\delta_{in}[m_{in}]$ gyártó döntés leképezés, $\delta_{out}[m_{out}]$ felhasználó döntés leképezés, ekkor a $\delta[m_{in} \cup m_{out}] = \delta_{in}[m_{in}] \cup \delta_{out}[m_{out}]$ **döntés leképezés** $m = m_{in} \cup m_{out}$ felett (mint a definíció is mutatja, az így definiált döntés leképezés nem függvény).

13. definíció. A $\delta_{in}[m]$ **gyártó döntés leképezés komplementere** a $\overline{\delta}_{in}[m] = \{(X, \Delta_{in}(X) \setminus \delta_{in}(X)) \mid X \in m\}$ leképezés.

14. definíció. A $\delta_{out}[m]$ **felhasználó döntés leképezés komplementere** a $\overline{\delta}_{out}[m] = \{(X, \Delta_{out}(X) \setminus \delta_{out}(X)) \mid X \in m\}$ leképezés.

15. definíció. A $\delta[m] = \delta_{in}[m_{in}] \cup \delta_{out}[m_{out}]$ **döntés leképezés komplementere** a $\overline{\delta}[m] = \overline{\delta}_{in}[m_{in}] \cup \overline{\delta}_{out}[m_{out}]$ leképezés.

16. definíció. A $\delta[m]$ döntés leképezés ($m \neq \varnothing$) akkor és csak akkor **konzisztens**, ha minden $X, Y \in m$-re $\delta(X) \cap \overline{\delta}(Y) = \varnothing$.

Jelölje $op(\delta[m])$ a $\delta[m]$ döntés leképezés műveleti egységeit, azaz $op(\delta[m]) = \{o \in O \mid o \in \delta(X)$ valamely $(X, \delta(X)) \in \delta[m]$-re$\}$, továbbá $mat(o) = \{x \in M \mid$ valamely $u \in o$ műveleti egységre $x \in in_u$ vagy $x \in out_u\}$.

17. definíció. Legyen $\delta[m]$ egy konzisztens döntés leképezés, $\mathbf{o} = op(\delta[m])$, $\mathbf{m} = mat(\mathbf{o}) \cup m$, és $\delta'[\mathbf{m}] = \{(X, Y) \mid X \in \mathbf{m}$ és $Y = \{i \in \mathbf{o} \mid X \in out_i\}\} \cup \{(X, Y) \mid X \in \mathbf{m}$ és $Y = \{i \in \mathbf{o} \mid X \in in_i\}\}$. Ekkor $\delta'[\mathbf{m}]$ döntés leképezés a $\delta[m]$ **döntés leképezés lezártja**. A $\delta[m]$ döntés leképezés zárt, ha $\delta[m] = \delta'[\mathbf{m}]$.

18. definíció. Két konzisztens döntés leképezés **ekvivalens**, ha a lezártjuk azonos.

A szükséges fogalmak bevezetése után most már megadhatjuk a döntés leképezés és a P-gráf kapcsolatát.

19. definíció. Adott az (m, o) P-gráf és a $\delta[m'] = \delta_{in}[m_{in}] \cup \delta_{out}[m_{out}]$ konzisztens döntés leképezés. Ha $op(\delta[m']) = o$, akkor $\delta[m']$ döntés leképezés az (m, o) **P-gráfhoz tartozó döntés leképezés**.

A fordított kapcsolat bemutatásaként tekintsük a $\delta[m']$ konzisztens döntés leképezést. Legyen $o = op(\delta[m'])$ és $m = mat(o) \cup m'$. Ekkor (i) az (m, o) P-gráf, (ii) $\delta[m']$ az (m, o) P-gráfhoz tartozó döntés leképezés. Ez alapján:

20. definíció. Egy $\delta[m]$ **konzisztens döntés leképezés P-gráfja** (m, o), ahol $o = op(\delta[m'])$, $m = mat(o) \cup m'$, és gráf($\delta[m']$)-mel jelöljük.

21. definíció. Legyenek $\delta_1[m_1]$ és $\delta_2[m_2]$ konzisztens döntés leképezések. A $\delta_1[m_1] = \delta_{1in}[m_{1in}] \cup \delta_{1out}[m_{1out}]$ döntés leképezés a $\delta_2[m_2] = \delta_{2in}[m_{2in}] \cup \delta_{2out}[m_{2out}]$ döntés leképezés **kiterjesztése** (jelölés: $\delta_1[m_1] \geq \delta_2[m_2]$), ha $m_{1in} \supseteq m_{2in}$ és $\delta_{1in}(X) = \delta_{2in}(X)$ minden $X \in m_{2in}$-re, továbbá $m_{1out} \supseteq m_{2out}$ és $\delta_{1out}(X) = \delta_{2out}(X)$ minden $X \in m_{2out}$-ra. A kiterjesztés reláció parciális rendezés a konzisztens döntés leképezések halmazán.

A szükséges előkészületi lépések után már definiálhatjuk az ABBW algoritmus szétválasztó lépését. Az alapesethez hasonlóan a korlátozó függvényekkel itt

sem foglalkozunk. A részproblémák definíciója, valamint a G korlátozó függvényre vonatkozó feltételek azonosak az alapesetben ismertetettekkel.

Az integrált hulladékkezelésnek köszönhetően egy adott részproblémát már nem kizárólag egy megfelelően kiválasztott anyag előállításáról hozott döntés alapján bonthatunk diszjunkt részproblémákra, hanem bizonyos esetekben anyagok különböző módon történő felhasználásáról hozott döntéssel is. Ennek megfelelően a döntési pontokként figyelembe vehető anyagokat két halmazzal adhatjuk meg.

Tekintsük a (P_r, P_p, R, I, O) hulladékkezeléssel integrált folyamat-hálózatszintézis feladat egy P_i - $S(\delta_i[m_i])$-vel adott - részproblémáját. A $\delta_i[m_i]$ döntés leképezés felírható mint $\delta_{i,in}[m_{i,in}] \cup \delta_{i,out}[m_{i,out}]$ valamely $m_{i,in} \subseteq m_i$ és $m_{i,out} \subseteq m_i$ halmazokra. Ekkor a P_i részprobléma a $p_{in}(S(\delta_i[m_i])) = ((mat^{in}(op(\delta_{i,out}[m_{i,out}])) \setminus mat^{out}(op(\delta_{i,out}[m_{i,out}]))) \cup P_r \cup P_p \cup mat^{in}(op(\delta_{i,in}[m_{i,in}])))\setminus (m_{i,in} \cup R)$ halmaz elemeinek gyártására, illetve a $p_{out}(S(\delta_i[m_i])) = (mat^{out}(op(\delta_{i,out}[m_{i,out}])) \cup (mat^{out}(op(\delta_{i,in}[m_{i,in}])) \setminus mat^{in}(op(\delta_{i,in}[m_{i,in}]))))\setminus m_{i,out}$ halmaz elemeinek felhasználására hozott újabb döntéssel bontható újabb részproblémákra, ahol mat^{out}, mat^{in} az MSGW algoritmusban is használt függvényeket jelöli.

Ha mindkét halmaz üres, az $S(\delta_i[m_i])$ részproblémában a megoldások struktúrája teljesen definiált, azaz egyértelmű, így a korlátozó függvény tulajdonságának megfelelően ennél a részproblémánál nincs szükség további szétválasztásra, a részprobléma levél. A valódi szétválasztást nem eredményező eseteket az alapesethez hasonlóan a maximális neutrális kiterjesztés alkalmazásával szűrhetjük ki.

22. definíció. A $\delta'[m'] = \delta[m] \cup \{(x,d)\}$ döntés leképezés a $\delta[m]$ konzisztens döntés leképezés **direkt neutrális kiterjesztése**, ha vagy (i) $x \in p_{in}(S(\delta[m]))$, $d \subseteq \Delta_{in}(x)$, és $\delta[m] \cup \{(x,c)\}$ inkonzisztens minden $c \in \wp(\Delta_{in}(x)) \setminus \{d\}$ esetén; vagy (ii) $x \in p_{out}(S(\delta[m]))$, $d \subseteq \Delta_{out}(x)$, és $\delta[m] \cup \{(x,c)\}$ inkonzisztens minden $c \in \wp(\Delta_{out}(x)) \setminus \{d\}$ esetén. A $\delta_n[m_n]$ döntés leképezés a $\delta_0[m_0]$ konzisztens döntés leképezés **neutrális kiterjesztése**, ha létezik konzisztens döntés leképezések egy $\delta_0[m_0]$, $\delta_1[m_1]$, …, $\delta_n[m_n]$ sorozata úgy, hogy $\delta_i[m_i]$ a $\delta_{i-1}[m_{i-1}]$ döntés leképezés direkt neutrális kiterjesztése (i=1, 2, …, n). A $\delta_{max}[m_{max}]$

döntés leképezés a $\delta[m]$ konzisztens döntés leképezés **maximális neutrális kiterjesztése**, ha neutrális kiterjesztése $\delta[m]$-nek, és $\delta_{max}[m_{max}]$-nak nincs direkt neutrális kiterjesztése. Az $S(\delta_{max}[m_{max}])=S(\delta[m])$ nyilvánvalóan teljesül.

Az ABBW algoritmus szétválasztó függvényének pontos definiálásához meg kell adnunk egy "anyagválasztó függvényt" ($x = A(S(\delta_i[m_i])) \in p_{in}(S(\delta_i[m_i])) \cup p_{out}(S(\delta_i[m_i]))$), ami lehet a legkisebb indexű anyag (a feladat definíció alapján), a legkevesebb módon előállítható illetve feldolgozható anyag, de ez lényegében tetszőleges, az ABBW algoritmus működését nem befolyásolja lényegesen.

Az ABBW algoritmus szétválasztó függvénye valamely $S(\delta_i[m_i])\neq\varnothing$ részproblémára:

$son(S(\delta_i[m_i])) = \{S(\delta_{ij}[m_{ij}]) \mid \delta_k[m_k] = \delta_i[m_i]\cup\{(x, c)\}, x = A(S(\delta_i[m_i])), c\in D,$ $\delta_k[m_k]$ konzisztens, δ_{ij} a δ_k maximális neutrális kiterjesztése$\}$,

ahol $D = \begin{cases} \wp(\Delta_{in}) & \text{ha } x \in p_{in}(S(\delta_i[m_i])) \text{ és } x \in P_p \setminus mat^{in}(op(\delta_i[m_i])) \\ \wp(\Delta_{in}) \setminus \{\varnothing\} & \text{ha } x \in p_{in}(S(\delta_i[m_i])) \text{ és } x \notin P_p \setminus mat^{in}(op(\delta_i[m_i])) \\ \wp(\Delta_{out}) \setminus \{\varnothing\} & \text{ha } x \in p_{out}(S(\delta_i[m_i])) \text{ és } x \in I \\ \wp(\Delta_{out}) & \text{ha } x \in p_{out}(S(\delta_i[m_i])) \text{ és } x \notin I \end{cases}$.

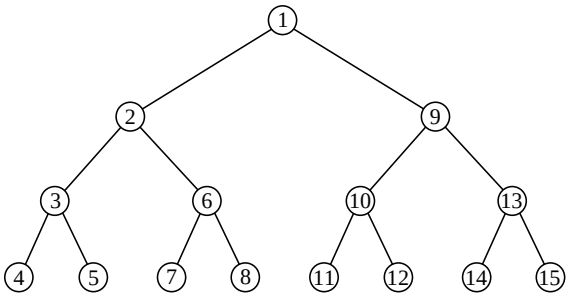
Ez a függvény megfelel mint szétválasztófüggvény, a feladat matematikai modelljétől függetlenül, minden hulladékkezeléssel integrált folyamat-hálózatszintézis feladat megoldására alkalmazható.

A döntés leképezés alapján egy adott részproblémánál polinomiális időben meghatározhatjuk a részproblémához tartozó összes megoldásstruktúrában szereplő (kiválasztott) műveleti egységeket, a kizárt műveleti egységeket és a választható műveleti egységeket. Ezek segítségével a műveleti egységek modelljét és ha szükséges, valamilyen relaxáló módszert felhasználva már meghatározható a korlátozó függvény.

3. példa (folytatás). A 18 műveleti egységet tartalmazó hulladékkezeléssel integrált folyamat-hálózatszintézis feladatot az ABBW algoritmus legrosszabb esetben 15 részprobléma megvizsgálásával oldja meg. A branch-and-bound keresőfát a 12. ábra mutatja, a részproblémákat a 3. táblázat tartalmazza.

Ha egy feladat költségfüggvénye nem bünteti a kibocsátható anyagok gyártását, azaz az alapfeladat csak a közbülső anyagok gyártásának tiltásával bővült, az

SW4 axióma módosításával tovább szűkíthető a keresési tér. Ebben az esetben ugyanis a megoldásstruktúra hulladékkezelésre szolgáló részeiben a kibocsátható anyagok további feldolgozása már szükségtelen, ilyen struktúra nem tartozhat az optimális megoldáshoz.



12. ábra. A branch-and-bound algoritmus keresőfája a 3. példa megoldásakor legrosszabb esetben.

3. táblázat. A 12. ábra részproblémáinak már kiválasztott és még kiválasztható műveleti egységei.

részprobléma	kiválasztott műveleti egységek	kiválasztható műveleti egységek
1	∅	1, 2, 4, 5, 6, 10, 11, 12, 13, 16
2	5, 12, 16	4, 10, 11, 13
3	5, 12, 13, 16	4, 10, 11
4	5, 10, 12, 13, 16	∅
5	4, 5, 11, 12, 13, 16	∅
6	5, 12, 16	4, 10, 11
7	5, 10, 12, 16	∅
8	4, 5, 11, 12, 16	∅
9	6, 13, 16	1, 2, 4, 10, 11
10	1, 6, 13, 16	4, 10, 11
11	1, 6, 10, 13, 16	∅
12	1, 4, 6, 11, 13, 16	∅
13	2, 6, 13, 16	4, 10, 11
14	2, 6, 10, 13, 16	∅
15	2, 4, 6, 11, 13, 16	∅

A megváltozott (SW4) axióma:

(SW4') Minden a struktúrában szereplő y műveleti egységre az alábbi feltételek közül legalább egy teljesül

(a) Létezik a struktúrában az y műveleti egységet reprezentáló csúcstól valamely (lehetséges) terméket reprezentáló csúcsig vezető út.

(b) Létezik utak egy sorozata úgy, hogy y műveleti egységet reprezentáló csúcs az első út eleje, az utolsó út vége egy (lehetséges) terméket reprezentáló csúcs. Mindegyik út első és utolsó eleme műveleti egységet reprezentáló csúcs, mindegyik útnak közös az eleje vagy a vége a következő úttal és az utakban nincs közös anyagot reprezentáló csúcs.

(c) Létezik a struktúrában az y műveleti egységet reprezentáló csúcshoz vezető út, amelynek eleje egy olyan x közbülső anyag, amelyből nem vezet út (lehetséges) termékhez és x-et valamely, az SW4a vagy SW4b feltételnek eleget tevő műveleti egység gyártja; az úton x és y között nincs kibocsátható anyag.

Ennek megfelelően az MSGW és az ABBW algoritmusok is kisebb módosítással alkalmassá tehetőek ennek a feladattípusnak a még hatékonyabb megoldására [15, 39].

10.2. Integrált folyamat-hálózat- és irányítórendszer tervezés

Az integrált folyamat-hálózat- és irányítórendszer tervezésekor a költségfüggvény minimalizálása mellett a rendszer irányíthatósága is alapvető fontosságú. Ilyenkor a rendszer dinamikáját és szabályozhatóságát lehetőség szerint a rendszer tervezésének lehető legkorábbi fázisában kell figyelembe venni.

A hagyományos megközelítésben a termelőrendszer és az ehhez tartozó irányítórendszer két, független lépésben történő megtervezését végzik, ami általában nem ad optimális megoldást. A feladat megoldására javasolt eddigi módszerek egyike sem integrálja a rendszer tervezésével az irányítórendszer tervezését. Az irányítórendszer tervezésekor ugyan módosíthatják a rendszer optimális működését leíró állapotváltozókat, de rendszer struktúrája ebben a lépésben már kötött. Így előfordulhat, hogy egy költségminimumhoz tartozó megoldás nem, vagy nagyon nehezen irányítható, azaz a megoldás értéktelen.

Az általunk javasolt módszerben már a folyamat-hálózatszintézis során figyelembe veszünk irányíthatósági szempontokat.

Az alapvető nehézség a két tervezési feladat eltérő jellegéből adódik: míg a folyamat-hálózatszintézis során műveleti egységek egy olyan halmazát keressük, amelyek megfelelő módon működtetve minimális költséggel állítják elő a termék(ek)et, azaz a tervezés ezen része statikus jellegű, addig az irányítórendszer a hálózat dinamikus állapotváltozásaihoz kapcsolódik, tehát az irányítórendszer tervezéséhez a rendszer dinamikáját kell leírni. Ugyanakkor a struktúra tervezése során egyszerű Boolean típusú strukturális irányíthatósági szempontokat már figyelembe vehetünk, melyekkel a struktúra állapotától függetlenül írhatjuk le annak irányíthatósági tulajdonságait [21].

A kidolgozott módszerben irányíthatósági szempontokat vettünk figyelembe, amely szabályozórendszer tervezésére alkalmas. Hasonló algoritmussal a megfigyelhetőség is vizsgálható [18], ez már alkalmas diagnosztikára is.

A folyamat-hálózatszintézis feladat megoldására kidolgozott ABB algoritmusban a struktúrákat P-gráfokkal reprezentáljuk. A strukturális irányíthatóság vizsgálatára kidolgozott módszerekben irányított gráf alapú modelleket használnak a műveleti egységek dinamikájának struktúrális leírására. A javasolt módszer e két eredményre alapozva oldja meg az integrált folyamat-hálózat- és irányítórendszer tervezés (IPCS) feladatot. A struktúrákat a P-gráf egy kiterjesztésével tudjuk modellezni.

10.2.1. Struktúra reprezentáció

Az IPCS feladat megoldásakor a műveleti egység definícióját is módosítanunk kell. Míg az alapesetben egy rendezett (in_i, out_i, m_i, k_i) négyessel írtuk le a műveleti egységeket, ebben az esetben minden műveleti egység egy rendezett (in_i, out_i, a_i, c_i, m_i, k_i) hatossal írható le, ahol int_i, out_i, m_i, k_i szerepe változatlan, in_i a műveleti egység inputjainak halmaza, out_i a műveleti egység outputjainak halmaza, m_i a műveleti egység működését leíró függvény, k_i a műveleti egység költségfüggvénye. Az a_i a műveleti egység lehetséges aktuátorainak (beavatkozó változó) halmaza, míg c_i az ún. "hatásfüggvény", amelynek értelmezési tartománya az aktuátorok és input anyagok halmazának uniója,

értékkészlete az output anyagok halmazának hatványhalmaza. Az IPCS feladat hatásfüggvénye a műveleti egységek hatásfüggvényeinek uniójaként definiálható.

Az IPCS feladat a műveleti egységek egy olyan részhalmazának megkeresése a megfelelő állapotváltozókkal és aktuátorokkal, amelyekből felépülő hálózat (megoldás) egyrészt az adott nyersanyagok segítségével a kért termékeket előállítja úgy, hogy a struktúrába felvett aktuátorok segítségével a termékek és a struktúra minden közbülső (termelt és felhasznált) anyaga irányítható, másrészt a hálózat költsége minimális. Ebben az esetben a költség mind a gyártási mind az irányítási költségeket tartalmazza.

Az IPCS feladat definiálásához elegendő a műveleti egységek, nyersanyagok és termékek halmazának megadása. Megengedett az anyagok és aktuátorok halmazának különálló definiálása is, bár felesleges olyan aktuátor definiálása, amely egyik műveleti egységben sem lehetséges, illetve valamely műveleti egységnél olyan aktuátort megadni, amelyet nem tartalmaz a külön megadott aktuátorhalmaz. Ugyanakkor az alap folyamat-hálózatszintézis feladattól megkülönböztetésképp az IPCS feladatokat a (P, R, O, A) négyessel adjuk meg.

A kibővített műveleti egységekből álló struktúrákat az alapesethez hasonlóan irányított páros gráffal reprezentálhatjuk, ezeket megkülönböztetésképp **CP-gráfnak** nevezzük.

A gráf csúcspontjait a műveleti egységek (ezeket az alapesethez hasonlóan vízszintes vonallal, ▬, jelöljük), anyagok (jelölésük változatlanul kör, ●) és aktuátorok (jelölésük tömör négyzet, ■) alkotják. A nyersanyagokat és a termékeket továbbra is megkülönböztetjük a többi anyagtól (▽, illetve ⊙). A gráf élei az anyagok és műveleti egységek közötti, valamint a hatásfüggvény által leírt kapcsolatok: egy *b* műveleti egység típusú csúcsból él vezet egy *a* anyag típusú csúcshoz, ha az *a* eleme *b* outputhalmazának ($a \in \text{out}_b$), illetve egy *a* anyag típusú csúcsból él vezet egy *b* műveleti egység típusú csúcshoz, ha *a* eleme *b* inputhalmazának ($a \in \text{in}_b$). A hatásfüggvény által leírt relációkat reprezentáló éleket szaggatott vonallal jelöljük a műveleti egységek és input-output anyagaik kapcsolatát jelölő folyamatos élektől megkülönböztetésképp. A

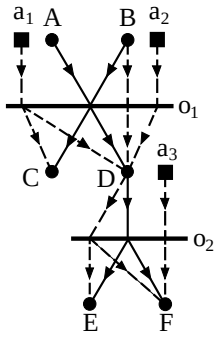
CP-gráfot a három különböző típusú csúcsainak halmazaiból képzett (M, A, O) rendezett hármassal definiáljuk.

4. példa. A 13. ábra egy két műveleti egységből, o_1 és o_2, álló CP-gráfot mutat be, ahol

$in_{o_1}=\{A, B\}$, $out_{o_1}=\{C, D\}$, $a_{o_1}=\{a_1, a_2\}$, $c_{o_1}=\{(B, \{D\}), (a_1, \{C, D\}), (a_2, \{D\})\}$

$in_{o_2}=\{D\}$, $out_{o_2}=\{E, F\}$, $a_{o_2}=\{a_3\}$, $c_{o_2}=\{(D, \{E\}), (a_3, \{F\})\}$.

A műveleti egységek állapotváltozói és költségfüggvényei a struktúra szempontjából lényegtelenek.



13. ábra. Az ({A, B, C, D, E, F}, {a_1, a_2, a_3}, {o_1, o_2}) CP-gráf.

Mint azt a CP-gráf definíciója mellett a fenti ábra is jól mutatja, a P-gráf része a CP-gráfnak.

23. definíció. Az (M, A, O) CP-gráf **strukturális komponense**, $(M, A, O)^S$, a CP-gráf aktuátor típusú csúcsainak és a hatásfüggvényt reprezentáló éleinek elhagyásával kapott P-gráf. Az (M, A, O) CP-gráf **kontrol komponense**, $(M, A, O)^C$, a CP-gráfból a műveleti egységek és anyagok közötti input-output relációkat leíró élek elhagyásával kapott gráf.

Az (M, A, O) CP-gráfban az aktuátorok halmaza részhalmaza a gráf O műveleti egységein lehetséges aktuátorok halmazának, ennek megfelelően a kontrol komponens élei is a műveleti egységek hatásfüggvényeinek csak egy részhalmazát reprezentálják.

24. definíció. Az a_1, a_2, ..., a_n sorozat **struktúra út** a CP-gráfban, ha út a strukturális komponensében, jelölése $[a_1, a_n]^S$. Ha b_1 aktuátor és b_1, b_2, ..., b_m út a CP-gráf kontrol komponensében, akkor **kontrol útnak** nevezzük és $[b_1, b_m]^C$-vel jelöljük.

A CP-gráfok csúcsai között több kontrol út is lehetséges, a CP gráf műveleti egységeinek és a kontrol út definíciójából következik, hogy aktuátor csak kontrol út elején lehetséges.

10.2.2. Kombinatorikusan lehetséges irányítható struktúrák

Az alapesethez hasonlóan a megoldásstruktúráknak bizonyos tulajdonságoknak meg kell felelniük. Ezeket a tulajdonságokat és az őket leíró axiómákat két csoportba oszthatjuk. Egy IPCS feladat megoldásainak kombinatorikusan lehetséges megoldásoknak kell lenniük: ha elhagyjuk az irányítórendszerre vonatkozó feltételeket, akkor egy alap folyamat-hálózatszintézis feladatot kapunk, tehát a megoldásstruktúráknak teljesíteniük kell az S1-S5 axiómákat. Az irányítórendszerre vonatkozó követelményt a feladatosztály definíciójában megfogalmaztuk: a struktúrába felvett aktuátorok segítségével a termékek és a struktúrában előforduló minden közbülső (termelt és felhasznált) anyag is irányítható. Ennek megfelelően a strukturális irányíthatóságot megfogalmazó axióma:

(SC1) A struktúrában minden x anyagra, amely termék vagy termelt és felhasznált anyag, létezik $a \in A$, hogy $[a, x]^C$ kontrol út a struktúra CP-gráfjában.

Formálisan: a (P, R, O, A) IPCS feladatra adott struktúra, amelyhez tartozó CP-gráf (M', A', O'), csak akkor lehet egy megoldás struktúrája, ha $\forall x \in P \cup (mat^{in}(O') \cap mat^{out}(O'))$-hoz $\exists a \in A'$, hogy $[a, x]^C$ kontrol út (M', A', O')-ban.

25. definíció. Az (M', A', O') CP-gráffal adott struktúra **kombinatorikusan lehetséges és irányítható**, röviden **CFC struktúra**, ha teljesíti az S1, S2, ..., S5 és SC1 axiómákat.

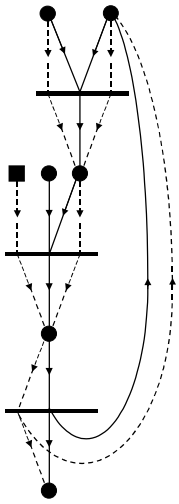
A 14. ábrán egy CFC struktúra, valamint egy kombinatorikusan lehetséges, de nem irányítható (azaz nem CFC) struktúra látható.

7. tétel. Egy IPCS feladat CFC struktúráinak halmaza zárt az unióra.

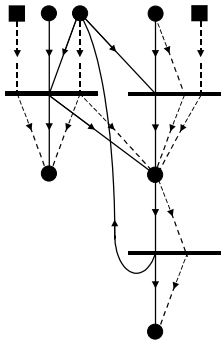
Bizonyítás: Legyen (M', A', O') és (M'', A'', O'') a (P, R, O, A) IPCS feladat két CFC struktúrájának CP-gráfja.

(i) A két struktúra uniója teljesíti az S1, S2, ..., S5 axiómákat: Az unió eredményeképp kapott CP-gráf strukturális komponense a két CP-gráf

strukturális komponensének uniója. Mivel két kombinatorikusan lehetséges P-gráf uniója is kombinatorikusan lehetséges [12], a két CP-gráf uniójaként kapott gráf strukturális komponense, így maga a CP-gráf is kombinatorikusan lehetséges.



14.a. ábra. CFC struktúra.



14.b. ábra. Kombinatorikusan lehetséges, de nem irányítható struktúra.

.

(ii) A két struktúra uniója teljesíti az SC1 axiómát: Az unió eredményeként kapott CP-gráf kontrol útjai az eredeti CP-gráfok kontrol útjainak uniója, azaz minden $x \in P \cup mat^{in}(O' \cup O'') \cap mat^{out}(O' \cup O'')$-re létezik $a \in A' \cup A''$, hogy $[a, x]^C$ kontrol út.

10.2.3. CMSG algoritmus

Az unióra zártság következményeként létezik maximális CFC struktúra. Az alapfeladathoz hasonlóan az optimális megoldás keresése leszűkíthető a CFC struktúrákra, így a maximális struktúra segítségével lényegesen csökkenthetjük a keresési teret. A maximális CFC struktúra egy hatékony polinomiális algoritmussal generálható. A maximális CFC struktúrát meghatározó CMSG algoritmust Pidgin Algol nyelven a 15. ábra mutatja.

10.2.4. Az algoritmus helyességének bizonyítása

Tekintsünk egy IPCS feladatot és a feladathoz tartozó maximális CFC struktúrát. Mivel az alap folyamat-hálózatszintézis feladat kombinatorikusan lehetséges megoldásstruktúráit leíró axiómák a CFC struktúrákat definiáló

Input: O*: a műveleti egységek halmaza*
M*: az anyagok halmaza*
P*: a termékek halmaza*
R*: a nyersanyagok halmaza*
A*: az aktuátorok halmaza*
Jelölések:
$o \subseteq O$, $mat^{in}(o)=\{x \in M \mid u \in o, x \in in_u\}$, $mat^{out}(o)=\{x \in M \mid u \in o, x \in out_u\}$
$m \subseteq M$, $op^{in}(m)=\{u \in O \mid m \in out_u\}$, $op^{out}(o)=\{u \in O \mid m \in in_u\}$
Azaz az op^{in}, mat^{in} *függvények egy csúcshalmazba bejövő élek kezdőpontjainak halmazát, míg az* op^{out}, mat^{out} *függvények egy csúcshalmazból kiinduló élek végpontjainak halmazát határozzák meg.*

procedure cmsg():
begin
st1: $O := O \setminus op^{in}(R)$;
rp1: **repeat**
st2: $M := mat^{in}(O) \cup mat^{out}(O)$; $r := mat^{in}(O) \setminus (mat^{out}(O) \cup R)$;
wh1: **while** r is not empty **do**
begin
$x \in r$; $M := M \setminus \{x\}$; $o := op^{out}(x)$; $O := O \setminus o$;
$r := (r \cup (mat^{out}(o) \setminus mat^{out}(O))) \setminus \{x\}$;
end;
co1: **if** $P \cap M \neq P$ **then stop**; **comment**: there is no maximal controllable structure
st3: $p := P$; $O_r := \emptyset$; $m := \emptyset$;
wh2: **while** p is not empty **do**
begin
$x \in p$; $m := m \cup \{x\}$; $o_x := op^{in}(x)$;
$O_r := O_r \cup o_x$; $p := (p \cup mat^{in}(o_x)) \setminus (R \cup m)$;
end;
st4: $r := \{x \mid \exists o_i \in O_r, \exists a \in a_i, x \in c_i(a)\}$; $s := \emptyset$; $O := O_r$;
wh3: **while** r is not empty **do**
begin
$x \in r$; $s := s \cup \{x\}$; $u := op^{out}(x)$;
for all $o_i \in u$ **do** $r := r \cup c_i(x)$; $r := r \setminus s$;
end;
co2: if $P \cap s \neq P$ **then stop**; **comment**: there is no maximal controllable structure
st5 $nco := op^{out}(mat^{in}(O_r) \setminus (s \cup R))$;
st6:: $O := O_r \setminus nco$; $M := mat(O)$;
until nco is empty;
st7: $M_r := mat^{in}(O_r) \cup mat^{out}(O_r)$; $A_r := \{x \in a_i \mid i \in O_r\}$; **write**(M_r, O_r, A_r);
end;

15. ábra. A CMSG algoritmus.

axiómák részhalmaza, az IPCS feladatból az irányíthatósági szempontokat elhagyva a maximális CFC struktúra strukturális komponense a kapott alap folyamat-hálózatszintézis feladatnak egy kombinatorikusan lehetséges megoldás struktúrája. Ebből következik, hogy a maximális CFC struktúra strukturális

komponense része az irányíthatósági feltételek elhagyásával kapott alap folyamat-hálózatszintézis feladat maximális struktúrájának. Ennek megfelelően az algoritmus működését az alábbi módon vázolhatjuk.

1. lépés. Az irányíthatósági feltételek elhagyásával határozzuk meg a (P, R, O) alapfeladat maximális struktúráját. Ha ez nem létezik, akkor a feladatnak nincs kombinatorikusan lehetséges megoldása és így nyilvánvalóan kombinatorikusan lehetséges és irányítható megoldása sincs.

2. lépés. Ha a kapott maximális struktúra irányítható, akkor ez egyben a maximális CFC struktúra is. Egyébként határozzuk meg az anyagok *s* halmazát, amelyekhez létezik kontrol út.

3. lépés. Ha a termékek valamelyike nem eleme az *s* halmaznak, nem létezik maximális CFC struktúra. Töröljük azokat a műveleti egységeket, amelyeknek valamely nem nyersanyag inputja (azaz a struktúrában gyártott és egyben felhasznált anyag) nem eleme az *s* halmaznak. Keressük meg a megmaradt műveleti egységekhez tartozó maximális struktúrát. Folytassuk az algoritmust a 2. lépéssel.

8. tétel. A wh3 ciklus befejeződésekor az *s* halmaz a műveleti egységek ($mat(O_r)$, O_r, $act(O_r)$) IPCS feladathoz tartozó struktúrában irányítható anyagokat tartalmazza.

Bizonyítás: Jelölje *c* az irányítható anyagok halmazát.

(i) $c \subseteq s$

Legyen $x \in c$, azaz létezik $[p_1, p_n]^C$ kontrol út, ahol p_1 aktuátor, $p_n = x$, minden $i=2, 3, ..., n$-re $\exists o_j \in O_r$, hogy $p_i \in c_j(p_{i-1})$. Az st3 utasítás után $p_2 \in r$. A ciklus i. futásakor a p_{i+2} bekerül az *r* halmazba, p_{i+1} pedig az *s* halmazba. A p_n anyag a ciklus n-1. futásakor kerül az *s* halmazba. Így a ciklus véges (a bővülő *s* halmazt kivonva az *r*-ből, valahány lépés után *r* üres lesz), *s* pedig tartalmazza p_2, p_3, ..., p_n anyagokat.

(ii) $c \supseteq s$

Az *s* halmaz konstrukciójából nyilvánvaló.

9. tétel. A CMSG algoritmus véges és polinomiális időben véget ér.

Bizonyítás: Mivel az algoritmus nem hív meg külső algoritmusokat és nem rekurzív, elegendő belátni, hogy a ciklusok végesek és minden ciklus lehetséges ismétléseinek száma polinomiális vagy az anyagok vagy a műveleti egységek számát tekintve.

A külső repeat ciklus ismételt futásának feltétele legalább egy műveleti egység törlése (st5) a maximális CFC struktúra lehetséges műveleti egységeinek halmazából, így ez a ciklus k műveleti egységgel definiált feladat esetén maximum k-1-szer ismétlődhet. A belső ciklusok közül a wh1 és wh2 ciklusok az MSG algoritmust részei, ezek végesek és polinom idejűek [12]. A wh3 ciklus az irányítható anyagokat határozza meg, lényegében egy gráfbejárás, ez nyilván véges, az anyagok száma mindig felső korlátot jelent a ciklus ismétlésének számára.

10. tétel. A CMSG algoritmus által meghatározott (M_r, O_r, A_r) struktúra CFC struktúra.

Bizonyítás:

(i) Az (M_r, O_r, A_r) struktúra kombinatorikusan lehetséges.

Az wh2 ciklus befejezésekor a (mat(O_r), O_r, act(O_r)) struktúra kombinatorikusan lehetséges (a CMSG algoritmus első része az MSG algoritmus). Ha az st5 utasításban az nco halmaz üres, az algoritmus befejeződik és az O_r halmaz nem változik.

(ii) Az (M_r, O_r, A_r) struktúra irányítható.

Mivel kaptunk maximális CFC struktúrát, a co2 feltétel nem teljesült és így minden termék irányítható. Az algoritmus befejeződésekor a $op^{out}(mat^{in}(O_r)\backslash(s\cup R)) = \varnothing$ (az nco halmaz üres), az op^{out} és a mat^{in} definíciójából következik, hogy ez csak akkor lehetséges, ha $mat^{in}(O_r)\backslash(s\cup R)) = \varnothing$, azaz $mat^{in}(O_r)\subseteq s\cup R$. Így $mat^{in}(O_r) \cap mat^{out}(O_r) \subseteq (s\cup R)\cap mat^{out}(O_r) = (s\cap mat^{out}(O_r))\cup(R\cap mat^{out}(O_r)) = s\cap mat^{out}(O_r) \subseteq s$. Mivel s az irányítható anyagok halmaza, a struktúra teljesíti a strukturális irányíthatóság axiómáját.

11. tétel. A CMSG algoritmus által meghatározott (M_r, O_r, A_r) struktúra a maximális CFC struktúra.

Bizonyítás: Megmutatjuk, hogy ha egy műveleti egység nem része az eredményként kapott struktúrának, akkor nem lehet része egyetlen megoldásnak sem.

(i) Ha egy műveleti egységet a maximális CFC struktúra lehetséges műveleti egységei közül az st1 utasítás zár ki, az nyersanyagként definiált anyagot állít elő, így az (S2) axióma szerint nem lehet megoldás része.

A további műveleti egységeket kizáró utasítások a ciklusokon belül vannak, ezért mindig azt kell megvizsgálnunk, hogy az O_r illetve O műveleti egység halmaz aktuális műveleti egységeit tekintve indokolt-e egy műveleti egység kizárása a további vizsgálatból.

(ii) Ha egy *b* műveleti egységet a wh1 ciklusban törlünk, akkor *b*-nek létezik az O műveleti egység halmazban nem előállított inputja, azaz nem lehet az O műveleti egység halmaz elemeiből képzett megoldásstruktúra része.

(iii) Ha egy *b* műveleti egység a wh2 ciklusban nem lesz eleme az O_r halmaznak, akkor nem adható meg út a *b* műveleti egység és valamely termék között az O műveleti egység halmaz aktuális műveleti egységeit felhasználva, így a ciklust követő st4 utasítás után az O műveleti egység halmaznak indokoltan nem lesz eleme.

(iv) Mint azt korábban bebizonyítottuk, a wh3 ciklus befejeződésekor az *s* halmaz az irányítható anyagok halmaza. Így az nco halmaz azon műveleti egységek halmaza, amelyeknek valamely inputja az O műveleti egység halmaz aktuális műveleti egységeivel nem irányítható. Így az nco elemei nem lehetnek az O műveleti egységeiből képzett megoldásstruktúrák részei.

12. tétel. Ha a CMSG algoritmus nem ad maximális CFC struktúrát, akkor a feladatnak nincs kombinatorikusan lehetséges és irányítható megoldása.

Bizonyítás: Az algoritmus szerint nincs maximális CFC struktúra, ha az algoritmus végrehajtása során egy O műveleti egység halmazra a co1 vagy co2 feltétel teljesül. Ekkor az O műveleti egység halmaz elemeiből nyilván nem adható meg megoldásstruktúra. A 11. tétel bizonyításában megmutattuk, hogy a műveleti egységek törlése mindig indokolt, így az inputként adott műveleti egység halmaz elemeiből sem adható meg megoldásstruktúra, így a feladatnak nem létezik megoldása.

10.2.5. IPCS feladat megoldása az ABB algoritmus kiterjesztésével

Amíg az algoritmusban a megoldások struktúrája nem teljesen meghatározott, azaz amíg egy adott részprobléma több megoldásstruktúrát reprezentál, az irányíthatósági szempontokat relaxáljuk, így az alap folyamat-hálózatszintézis feladat részprobléma reprezentációját és szétválasztó függvényét változtatás nélkül alkalmazhatjuk. Mivel a strukturális komponens és a kontrol komponens költségei additívek, az így kapott érték az IPCS feladat esetén is megfelel mint költségfüggvény. Azaz a költségfüggvényt a következő módon definiálhatjuk:

Közbülső részproblémák esetén, ha egy részprobléma nem tartalmaz kombinatorikusan lehetséges és irányítható struktúrákat, akkor kizárhatjuk a további vizsgálatból, az alapfeladat költségfüggvényének változtatás nélküli alkalmazása előtt egy egyszerű gráfbejáró algoritmussal ellenőrizve a strukturális irányíthatóságot.

Teljesen definiált strukturális komponens esetén valamely hagyományos módszer segítségével meghatározzuk az irányítórendszerrel integrált folyamat-hálózatszintézis feladat egy lehetséges struktúrájának optimális működését, azaz az állapotváltozókat és a struktúrába felvett aktuátorok halmazát.

10.3. Folyamat-hálózatszintézis szakaszosan folytonos költségfüggvényű műveleti egységekkel

Mint korábban jeleztük, nagy ipari feladatok megoldása esetén a hagyományos branch-and-bound algoritmust alkalmazva nagyon sok részproblémát kell megoldani, amelyek mindegyike egy matematikai programozási feladat megoldását jelenti. Ennek megfelelően az eddigi módszerek a műveleti egységek költségfüggvényéről a jobb megoldhatóság érdekében általában feltételezik, hogy az folytonos és konkáv függvény. A gyorsított branch-and-bound algoritmus segítségével ezen részproblémák számát nagyságrendekkel csökkentettük, így lehetővé vált, hogy bonyolultabb, a valóságot jobban modellező költségfüggvényt használjunk.

A gyakorlatban a műveleti egységek csak adott méretekben elérhetőek, ezt figyelembe véve egy adott típusú műveleti egység költségfüggvénye csak szakaszosan folytonos. A szakaszosan folytonos költségfüggvényű műveleti

egységekből álló folyamat-hálózatszintézis megoldására kidolgozott módszer a gyorsított branch-and-bound algoritmuson alapul. A branch-and-bound módszer szétválasztó lépésének kiterjesztésével, a műveleti egységek kapacitására alsó és felső korlát számolásával megoldottuk a szakaszosan folytonos költségfüggvény miatt felmerülő problémákat.

1.1.1. Szakaszosan folytonos költségfüggvény kezelése

Szakaszosan folytonos költségfüggvényű műveleti egységek folyamat-hálózatszintézisének megoldására kidolgozott módszerünkben a költségfüggvény folytonos szakaszairól feltételezzük, hogy azok nemcsökkenő konkáv függvények. Ez nem jelent komoly megszorítást, mivel ez a feltétel szinte mindig teljesül gyakorlati példák esetében.

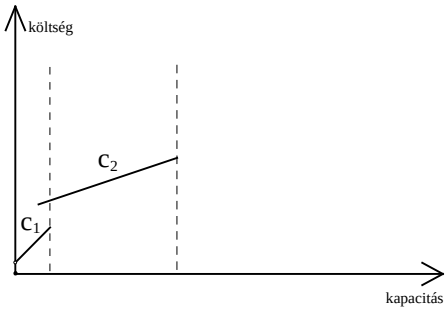
Tekintsük a folyamat-hálózatszintézis feladat megoldása során a költségfüggvény szakaszos folytonossága miatt felmerülő nehézségeket és azok megoldását a kidolgozott módszerben.

1. probléma. Ha egy műveleti egység több méretben elérhető, bizonyos esetekben kifizetődőbb lehet kisebb műveleti egységek valamely kombinációját alkalmazni, ha ez megengedett.

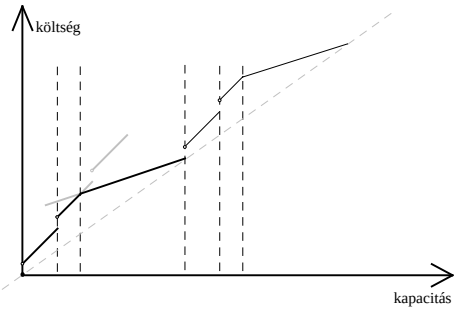
Az esetlegesen kifizetődőbb műveleti egység kombinációk ill. műveleti egység többszörözések meghatározása a folyamat-hálózatszintézis feladatot megoldó algoritmus indítása előtt megoldható, így ezzel a problémával az algoritmusban már nem kell foglalkoznunk. Miután megoldottuk a feladatot, a műveleti egységek optimális működéseként kapott kapacitásértékekből megállapíthatjuk, hogy az adott kapacitáshoz műveleti egységek milyen kombinációja vagy többszörözése tartozik. A különböző műveleti egységek kombinációjának illetve valamely műveleti egység többszörözésének feltétele, hogy az "összetett műveleti egység" költségfüggvényét illetve állapotváltozóit meg tudjuk adni, a gyakorlatban a költségfüggvényre az additivítás szinte mindig teljesül.

5. példa. Tegyük fel, hogy egy műveleti egység két különböző méretben létezik. Mindkét műveleti egységhez adott a költségfüggvény, jelöljük ezeket c_1 illetve c_2-vel, amelyekről itt az egyszerűbb ábrázolás kedvéért feltételezzük, hogy lineárisak és a műveleti egységeket kombinálva összeadódnak. A két műveleti

egységhez tartozó kapacitás-intervallumok egymást átfedhetik, a műveleti egységek kombinációja és többszörözése megengedett. Az eredeti és az algoritmus inicializáló lépésében meghatározható költségfüggvényt a 16. ábra mutatja.



16.a. ábra. Egy műveleti egység több típusának költségfüggvényei.



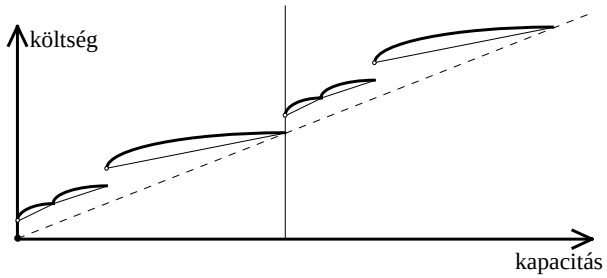
16.b. ábra. Egy műveleti egység költségfüggvénye az inicializáló lépés után.

2. probléma. Egy szakaszosan folytonos függvény se nem folytonos, se nem konkáv.

A megoldó algoritmusban két relaxált költségfüggvényt vezettünk be, illetve a szétválasztó függvényben figyelembe vesszük a költségfüggvény folytonos részintervallumait. Mindig meg tudunk adni egy a költségfüggvényt alulról közelítő lineáris függvényt a folytonos intervallum kiválasztása nélkül is (a szaggatott vonal a 17. ábrán), mivel a lehetséges műveleti egységek kapacitásainak felső korlátja véges, így az egy műveleti egységhez tartozó maximális kapacitásérték feletti (függőleges vonal a 17. ábrán) kapacitás elérése már csak műveleti egységek többszörözésével lehetséges. A módszer hatékonyságának javítására egy második, élesebb korlátot adó relaxált célfüggvényt is bevezethetünk, amely már egy adott kapacitás-intervallum kiválasztása után közelíti a célfüggvény egy folytonos szakaszát, így NLP feladatot megoldó korlátozó függvényt elegendő csak teljesen meghatározott struktúrák esetén használni, közbülső részproblémáknál elegendő egy LP feladatot megoldó korlátozó függvény alkalmazása.

Természetesen nagyon fontos megfelelő, az NLP feladatot hatékonyan megoldó, módszer kiválasztása is, mivel a levélfeladat megoldása egy nem hatékony módszerrel több időt igényelhet, mint a feladat többi részeinek megoldása.

A 17. ábra egy műveleti egység szakaszosan folytonos költségfüggvényét és relaxált függvényeit mutatja be.



17. ábra. Egy műveleti egység szakaszosan folytonos célfüggvénye.

3. probléma. Általában nem ismert alsó, illetve felső korlát a műveleti egységek kapacitására.

A javasolt módszer szétválasztó lépésében éles alsó és felső korlátokat határozunk meg. A kapacitáskorlátok számolására két különböző módszert alkalmazhatunk együttesen.

Az **analitikus módszer** LP feladatok megoldását jelenti. Egy műveleti egység felső kapacitáskorlátjának kiszámolásához az alábbi LP feladatot kell megoldani:

maximalizáljuk a vizsgált műveleti egység kapacitását

feltéve, hogy

az anyagegyensúly teljesül,

a relaxált költségfüggvény ≤ az optimális megoldás felső korlátja.

Az optimális megoldás nyilván felső korlát lesz a műveleti egység optimális megoldásban lehetséges kapacitására. Az alsó korlát hasonló módon állapítható meg:

minimalizáljuk a vizsgált műveleti egység kapacitását

feltéve, hogy

az anyagegyensúly teljesül,

a relaxált költségfüggvény ≤ az optimális megoldás felső korlátja.

A második feltétel elhagyásával kapott eredmény szintén valódi alsó kapacitáskorlátot eredményezne, de bizonyos esetekben az így kapott korlát élesebb.

Amint az látható, a kapacitáskorlátok analitikus számolásához szükségünk van az optimális megoldás költségének felső korlátjára. Ezt megkaphatjuk, ha a kezdő lépésben a korlátozó függvény által meghatározott megoldás költségét az eredeti költségfüggvénybe behelyettesítve is meghatározzuk. Ha ezt a korlátozó függvény minden hívása után megtesszük, a korlátot élesebbé tehetjük.

A **kombinatorikus módszer** egy egyszerű kereslet-ellenőrzés a vizsgált műveleti egység input-output anyagaira. A módszer csak abban az esetben alkalmazható, ha a műveleti egységek input-output anyagai között valamilyen arányokat meg tudunk állapítani, illetve lineáris függvénnyel jól közelíthetjük az input-output anyagok mennyiségét leíró összefüggéseket.

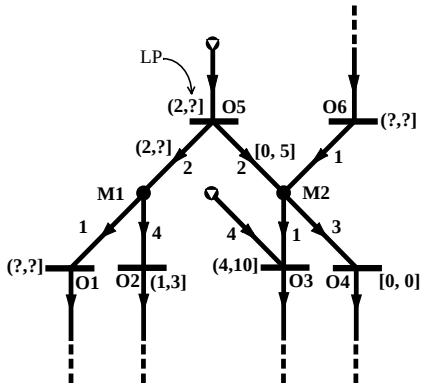
6. példa. Tekintsünk egy folyamat-hálózatszintézis feladat megoldásának egy közbülső fázisát. Az 18. ábra a feladat P-gráfjának egy részét mutatja. Az élekre írt számok az anyagok ki, illetve bemenő arányát adják meg a műveleti egységekre.

Tegyük fel, hogy az aktuális részproblémában az O2 és az O3 műveleti egységeket kiválasztottuk, az O4 műveleti egységet pedig kizártuk egy korábbi döntéssel valamely anyagok gyártására, továbbá az O2 és O3 műveleti egységek kapacitása az (1, 3] illetve a (4, 10] intervallumon belül változhat. Jelenleg az M1 anyag gyártására az O5 műveleti egységet választottuk ki. Az O5 műveleti egység kapacitására alsó és felső korlát számolásához a műveleti egység input és output anyagait kell megvizsgálnunk.

A műveleti egység egyetlen inputja nyersanyag, ha a nyersanyag korlátozott mennyiségben áll rendelkezésre, akkor ez felső korlátot jelent a műveleti egység kapacitására, egyébként az input vizsgálata nem ad felső korlátot. Input anyagok mennyiségéből alsó korlát nem határozható meg, mivel ha nem szükséges, nem kell feldogozni őket.

Az M1 output anyagból legalább 4 egység előállítása szükséges (O2 műveleti egység inputjaként), mivel csak az O5 műveleti egység állítja elő az M1 anyagot, az O5 műveleti egység kapacitása legalább 2. Az M1 anyagot vizsgálva felső korlátot nem tudunk megadni, mivel az O1 műveleti egység felső korlátja ismeretlen és így az M1 anyagból szükséges mennyiség is. Az M2 output anyagot vizsgálva alsó korlát nem adható meg, mivel az M2 anyagot az

O6 műveleti egység is gyártja és ennek a műveleti egységnek még nem ismert a felső kapacitás-korlátja. Az M2 anyagból legfeljebb 10 egység előállítása szükséges, így az M2 anyagot vizsgálva a felső korlát az O5 műveleti egység kapacitására 5. Tehát az input anyagok vizsgálata nem eredményez kapacitáskorlátot, míg az output anyagok vizsgálata az O5 műveleti egység alsó kapacitáskorlátjára 2-t ad. A felső kapacitáskorlátot az analitikus módszerrel határozhatjuk meg.



18. ábra. Alsó-felső korlát kombinatorikus meghatározása.

A javasolt algoritmusban, ha lehetséges mindkét módszert alkalmazzuk. A kombinatorikus módszer nem mindig ad kapacitáskorlátot, mivel bizonyos esetekben a műveleti egység nem minden input-output anyagára adható meg maximális igény (minimális igény nullával becsülhető), ugyanakkor egyszerűsége miatt érdemes alkalmazni, akár az analitikus korláttal megegyező eredményt is adhat. Az alsó és felső kapacitáskorlátokat meghatározó eljárásokat a javasolt branch-and-bound módszer szétválasztó lépésének leírásában adjuk meg.

4. probléma. A feladat mérete nagyon nagy lehet a bináris változók számát tekintve. Mivel a módszer az ABB algoritmusra épül, a kombinatorikus jellegből adódó gyorsítási lehetőségeket kihasználjuk, a javasolt módszerrel viszonylag nagy, hagyományos módszerrel nagyon sok bináris változót tartalmazó feladat is megoldható. Az összetettebb szétválasztó lépés bevezetése nyilván lényegesen növeli a megoldandó részproblémák számát, azonban a szétválasztó lépés sok esetben nem megoldható részproblémát eredményez.

10.3.2. Branch-and-bound algoritmus szakaszosan folytonos költségfüggvényű műveleti egységekkel adott folyamat-hálózatszintézis feladat megoldására

Az eddig bemutatott kiterjesztésekben a részprobléma reprezentációra megfelelő volt a döntés leképezés. Ebben az esetben nem csak valamely anyagot gyártó műveleti egységeket határozzuk meg, hanem a műveleti egység költség-függvénye, valamint a műveleti egységre kiszámolt alsó és felső korlát alapján kiválasztunk a műveleti egység kapacitására egy olyan intervallumot is, amelyen a műveleti egység költségfüggvénye folytonos. Ezért egy P_i részprobléma definiálására az $\delta_i[m_i]$ döntés leképezés mellett egy rendezett párokból álló I_i halmazt használunk, amelyben a párok első eleme egy műveleti egység, a második elem a műveleti egység költségfüggvényének egy folytonos szakasza.

I_i = {(u, (x, y]) | $u \in op(\delta_i[m_i])$, (x, y] az u műveleti egység költségfüggvényének egy folytonos szakasza}

Az I_i tekinthető egy olyan függvénynek, amely az $op(\delta_i[m_i])$-beli műveleti egységekre megadja a korábbi döntésekkel kiválasztott folytonos szakaszt. A részprobléma által reprezentált megoldások halmazát így $S(\delta_i[m_i], I_i)$-vel jelöljük és azon megoldásokat tartalmazza, amelyekhez tartozó P-gráf az aktuális részprobléma döntés leképezése által definiált P-gráf részgráfja és nem tartalmaznak az eddigi döntésekkel kizárt műveleti egységeket, továbbá műveleti egységek kapacitása az I_i-ben adott intervallumon belül van.

Ha valamely részproblémára igaz, hogy minden a részproblémába tartozó megoldásnak azonos a struktúrája, továbbá a műveleti egységek kapacitásai mindig ugyanabba a folytonos szakaszba esnek, akkor a branch-and-bound algoritmusban egy részproblémára meghatározott korlátozó függvénynél megköveteljük, hogy az F és G függvények értékei megegyezzenek, így az ilyen részproblémákra további szétválasztó lépések nem szükségesek.

A lehetséges döntési pontok halmaza, a $p(S(\delta_i[m_i], I_i))$ halmaz, azonos az alapesettel, nem függ az I_i halmaztól. A $p(S(\delta_i[m_i], I_i))$ halmazból az *m* döntési pontot kiválasztó $A(S(\delta_i[m_i], I_i))$ anyagválasztó függvény is változatlan, a szétválasztó lépésben az *m* anyagot gyártó műveleti egységek összes konzisztens kiválasztása mellett az I_i halmazba nem tartozó műveleti egységek

méretét is meg kell határozni a műveleti egységek kapacitására meghatározott alsó és felső korlát alapján lehetséges összes kapacitásintervallumot kiválasztva. Az *m* anyagot gyártó *b* műveleti egységekre az alsó, illetve felső korlátot az *lb*(S($\delta_i[m_i]$, I_i), *b*), illetve az *ub*(S($\delta_i[m_i]$, I_i), *b*) függvények segítségével adjuk meg. Ezek a korábban vázolt analitikus és kombinatorikus módszereket alkalmazva számolják a korlátokat, a függvényeket minden olyan *m*-et előállító *b* műveleti egységre meghívjuk, amely nem eleme op($\delta_i[m_i]$)-nek. A függvényeket megvalósító algoritmusok Pidgin Algol nyelvű változatát a 19. ábra mutatja be.

Az alapesethez hasonlóan előfordulhat, hogy bizonyos anyagokra nem történik valódi szétválasztás. Ezeket az eseteket az alapesetben is alkalmazott maximális neutrális kiterjesztéshez hasonló módon kezeljük. Megkeressük azokat a döntési pontokat, amelyeknél csak egy konzisztens előállítási mód létezik és a műveleti egységek kapacitásintervalluma is kötött (mert már elemei az I_i halmaznak, vagy a kapacitáskorlátként kapott alsó és felső korlátok egy folytonos szakaszon belülre esnek a célfüggvényükben), ezeket az eddigi döntésekhez hozzávéve a korlátozó függvény kiszámolása előtt, az újabb szétválasztó lépésekben mindig valódi szétválasztást fogunk végezni. A szétválasztó függvény ezek alapján valamely nem üres S($\delta_i[m_i]$, I_i) részproblémára:

son(S($\delta_i[m_i]$, I_i)) = {S($\delta_{ij}[m_{ij}]$, I_{ij}) | $\delta_k[m_k]$ = {($\delta_i[m_i] \cup \{(x, c)\}$, x = A(S($\delta_i[m_i]$, I_i)), $c \in \wp(\Delta(x)) \setminus \{\varnothing\}$, $\delta_k[m_k]$ konzisztens, $I_k = I_i \cup \{(b, (e, f]) \mid b \in c$, (e, f] intervallum b költségfüggvényének egy folytonos szakasza, az [lb(S($\delta_i[m_i]$, I_i), b), ub(S($\delta_i[m_i]$, I_i), b)] intervallumon belül}, ($\delta_{ij}[m_{ij}]$, I_{ij}) a ($\delta_k[m_k]$, I_k) maximális neutrális kiterjesztése}

A korlátozó függvény lényegében változatlan, a korlátozó feltételek közé a kapacitáskorlátokat is fel kell venni és kétféle relaxált költségfüggvénnyel dolgozhatunk.

7. példa. Tekintsünk egy hat műveleti egységből álló folyamat-hálózatszintézis feladatot, a műveleti egységek költségfüggvénye szakaszosan folytonos. A feladat két termék gyártása minimális költséggel négy nyersanyag segítségével.

```
Input:   δ[m]: a branch-and-bound ezen ágán eddig meghozott döntések
         b: az aktuális szétválasztó lépésben vizsgált műveleti egység
         I: a b műveleti egység kapacitására vonatkozó korlátok (intervallum)
Jelölések:
         (nem vezettünk be új jelölést az eljárás leírásában)
procedure lb(δ[m], I, b):
begin
lb := 0;
for all output material m of b do
     begin
     if x∉Δ[m] produces m   // x műveleti egység lehet S(δ[m], I)-beli megoldás része,
                            // de kapacitására nem ismert felső korlát
     then lbm := 0
     else lbm := ("min. demand"-"max. production")/ratio;
     // az m anyagból minimálisan szükséges, illetve maximálisan termelt mennyiség a δ[m] és
     // I paraméterek segítségével könnyen meghatározható
     if lbm>lb then lb := lbm;
     end
x := {min. b kapacitása | relaxált költségfüggvény ≤ aktuális optimum,
     anyagegyensúly} // LP
if x>lb then return x;
return lb;
end


procedure ub(δ[m], I, b):
begin
ub := 0;
for all output material m of b do
     begin
     if x∉Δ[m] consumes m   // x műveleti egység lehet S(δ[m], I)-beli megoldás része,
                            // de kapacitására nem ismert felső korlát
     then ubm := ∞
     else ubm := ("max. demand"-"min. production")/ratio;
     // az m anyagból maximálisan szükséges, illetve minimálisan termelt mennyiség a δ[m] és
     // I paraméterek segítségével könnyen meghatározható
     if ubm>ub then ub := ubm
     end
x := {max. b kapacitása | relaxált költségfüggvény ≤ aktuális optimum,
     anyagegyensúly} // LP
if x<ub then return x;
return ub;
end
```

19. ábra. A részproblémákra alsó illetve felső korlátot számoló függvények.

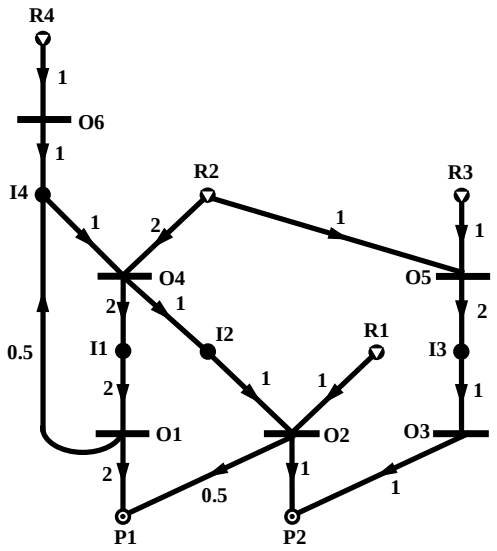
Feltételezhetjük, hogy nincs felső korlát a rendelkezésre álló nyersanyagokra. A feladat maximális struktúrája mind a hat műveleti egységet tartalmazza, a P-

gráfot a 20. ábra mutatja. A műveleti egységek költségfüggvényeit a 4. táblázat tartalmazza, a műveleti egységek modelljében az input output arány fix, a műveleti egység egyetlen állapotváltozója a műveleti egység kapacitása. A műveleti egységek kombinációja és többszörözése megengedett, az inicializáló lépésben kiszámolt költségfüggvényt és a relaxált függvényeket a 21. ábra tartalmazza.

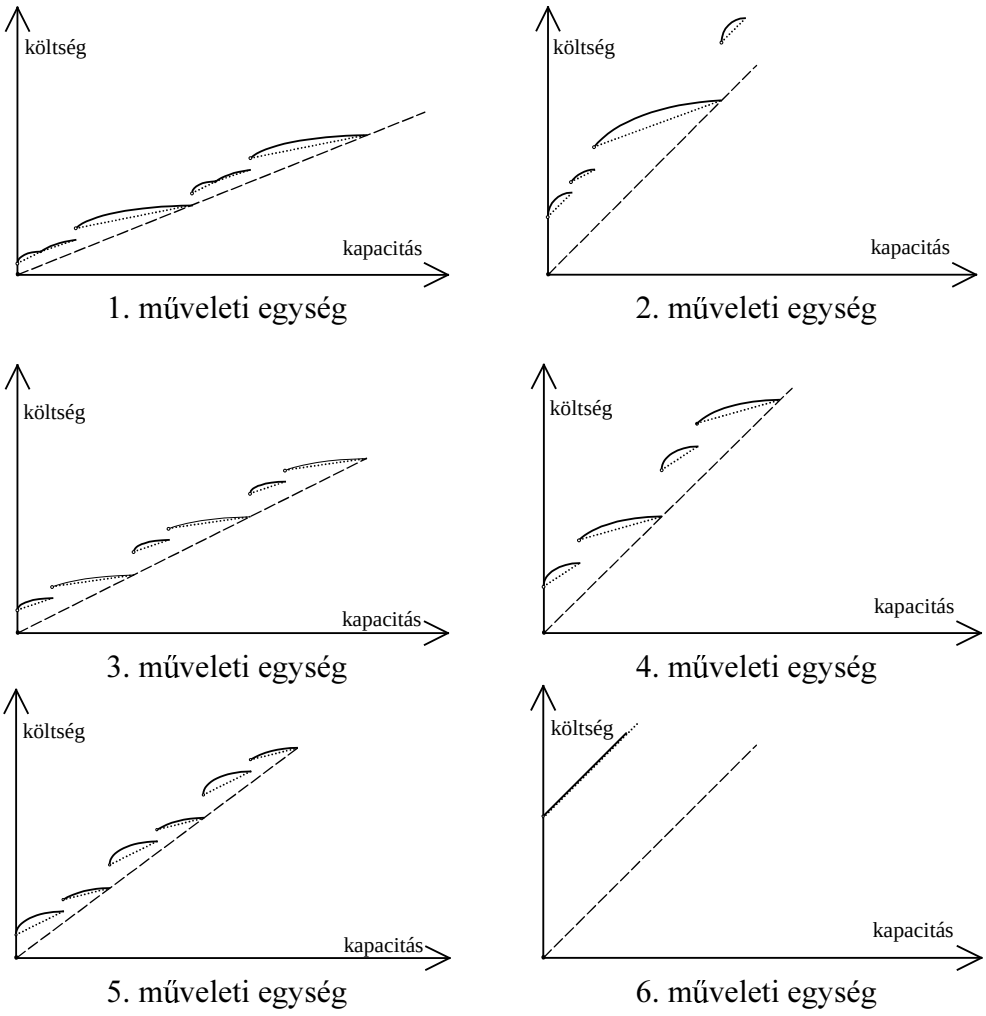
Hagyományos módszerrel ez a feladat 16 bináris változóval reprezentálható, a költségfüggvény nemlineáris. Az optimális megoldást 42 LP és 2 NLP részprobléma megoldása után találtuk meg, ez az O1, O3, O4, O5 és O6 műveleti egységeket tartalmazza, a hozzájuk tartozó kapacitásvektor (3, 0, 6, 3, 3, 1.5).

4. táblázat. A 7. példa műveleti egységeinek költségfüggvénye.

Műveleti egység típusa	Kapacitás-korlát	Költség-függvény	Relaxált függvény az intervallumra	A műveleti egység típus relaxált költségfüggvénye
1	0	0	0	0.4x
1	$0<x\leq2$	$0.66x^{0.6}+1$	0.5x+1	
1	$2<x\leq5$	$0.9x^{0.6}+0.64$	0.33+1.33	
1	$5<x\leq15$	$0.81x^{0.6}+1.9$	0.2x+3	
2	0	0	0	x
2	$0<x\leq2$	$1.32x^{0.6}+5$	x+5	
2	$2<x\leq4$	$1.32x^{0.6}+6$	0.5x+7	
2	$4<x\leq15$	$1.44x^{0.6}+7.7$	0.36x+9.54	
3	0	0	0	0.5x
3	$0<x\leq3$	$0.5x^{0.6}+2.1$	0.33x+2	
3	$3<x\leq10$	$0.5x^{0.6}+3.1$	0.14x+3.57	
4	0	0	0	x
4	$0<x\leq3$	$x^{0.6}+4.1$	0.66x+4	
4	$3<x\leq10$	$x^{0.6}+6.1$	0.28x+6.85	
5	0	0	0	0.75x
5	$0<x\leq4$	$0.88x^{0.6}+2$	0.5x+2	
5	$4<x\leq8$	$0.88x^{0.6}+3$	0.25x+4	
6	0	0	0	x
6	$0<x\leq2$	x+12	x+12	



20. ábra. A 7. példa maximális struktúrája.



21. ábra. Az inicializáló lépésben kiszámolt költségfüggvény és két relaxációja. (Jelölések: folytonos vonal – költség, szagatott vonal – kezdeti relaxáció, pontozott vonal – relaxáció az intervallum kiválasztása után.)

10.4. Folyamat-hálózatszintézis feladat megoldása párhuzamos működési elvű számítógépeken

Az eddigiektől eltérően az itt bemutatott módszer nem az alap folyamat-hálózatszintézis feladat egy kiterjesztését oldja meg, hanem az alap folyamat-hálózatszintézis feladatot megoldó ABB algoritmus technikai jellegű kiterjesztését, annak párhuzamos működési elvű számítógépeken végrehajtható változatát mutatjuk be.

Mivel csak az algoritmus megvalósítása tér el, ebben az esetben a struktúra reprezentáció, a gyorsított branch-and-bound szétválasztó-, illetve költségfüggvénye is változtatás nélkül alkalmazható. Amíg azonban a szekvenciális algoritmus egyben megadja a számítógép által végrehajtandó utasítássorozatot, párhuzamos működési elvű számítógép esetén minden egyes processzorra meg kell adni, hogy az algoritmus mely részét, mely utasítássorozatot kell végrehajtania.

Természetesen megengedett, hogy ugyanazt az algoritmusrészt több processzor is végrehajtsa, sőt az is lehetséges, hogy az összes processzor feladata azonos. Mivel a párhuzamosan működő processzorok ugyanazon feladat megoldásán dolgoznak, általában annak független részfeladatait végrehajtva, a processzorok a folyamat-hálózatszintézis feladatmegoldását szolgáló utasítások mellett a feladatok végrehajtását vezérlő processzorok közötti kommunikációt is végeznek. Közös memóriaterülettel is rendelkező rendszerek esetén ez jelentheti valamilyen adat felírását a közös memóriaterületre. Osztott memóriájú rendszerek esetén valódi kommunikációról van szó, a processzorok csatornákon keresztül üzeneteket küldve vezérlik a feladat végrehajtását. A fejezetben ismertetett algoritmus is egy osztott memóriájú rendszerre, **transzputer**re készült.

Ilyen esetekben egy processzor gyakran több részfeladatot – processzt - futtat, ezek egyike általában a feladatot megoldó processz, itt például az ABB algoritmus, míg egy másik processz például az üzenetek kezelését (fogadását, továbbítását) végzi.

A feladat megoldása szempontjából felesleges, kommunikációt végző processz általában jóval egyszerűbb, a gépidőnek csak a töredékét használja (különben

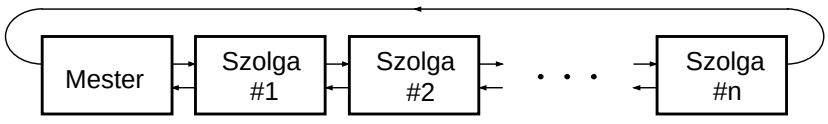
nem érünk el gyorsulást a több processzor használatával). Ugyanakkor léteznek olyan módszerek, amikor valamelyik processzor alapvető feladata a megoldás vezérlése, azaz a kommunikáció irányítása, de itt a többi processzornál a kommunikáció valóban elhanyagolható, ezek az ún. mester-szolga típusú algoritmusok.

A branch-and-bound típusú algoritmus párhuzamos végrehajtására két alapvető módszer létezik: az egyik a mester-szolga (master-slave) algoritmus, a másik a dinamikus processzorterhelés-elosztó (load balance) módszer.

10.4.1. Szoftver-architektúra

Mivel folyamat-hálózatszintézis feladat megoldása során egy adott részproblémából generált részproblémák száma nagy eltéréseket mutathat, nehéz minden feladattípusra jól működő "load balancing" módszert megadni. Ugyanakkor a folyamat-hálózatszintézis feladat egy-egy részproblémájának feldolgozása önmagában is nehéz feladat (korlátozó függvény kiszámolása, új részproblémák generálása), így mester-szolga algoritmust alkalmazva [38] várhatóan sok processzort használva sem válik a mester a feladat megoldása során szűk keresztmetszetté.

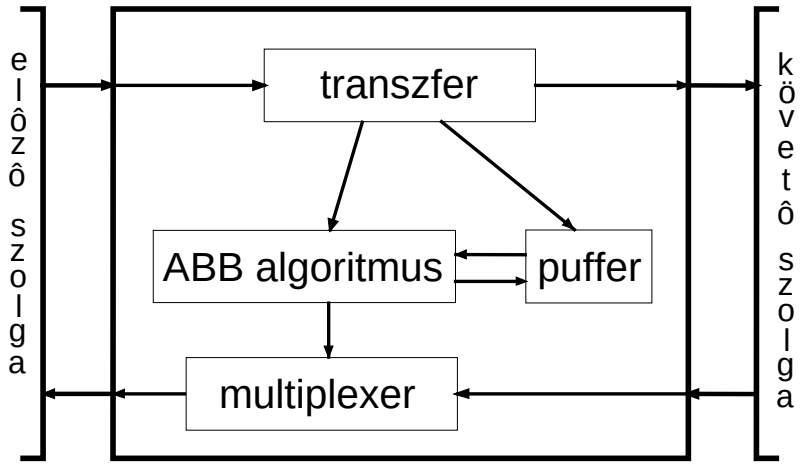
A processzorok közötti fizikai kapcsolat korlátozott, a mester-szolga típusú algoritmusoknál ideális csillag topológia helyett a számítógép struktúrájának megfelelő topológiát kell definiálni. Elkészült a párhuzamos ABB algoritmus alapváltozata gyűrű (22. ábra) és bináris fa processzortopológia esetén. Az algoritmusokat occam nyelven implementáltuk, ezt a konkurens algoritmusokhoz kifejlesztett nyelvet lényegében a transzputerrel együtt fejlesztették ki. (Transzputerhez való hozzáférést a JATE biztosította, amit ezúton is köszönünk.)



22. ábra. Gyűrű topológia.

Mint az a gyűrű topológiát bemutató ábrán is látható, a fizikai kapcsolat korlátai miatt a szolga processzeknek a többi processz közötti üzeneteket is továbbítania

kell. Ezért a szolga processz a részfeladatokat megoldó processz mellett több, kommunikációt biztosító processzt is tartalmaz (23. ábra).



23. ábra. Egy közbülső szolga struktúrája.

10.4.2. Eredmények

A kidolgozott eszközt három feladat megoldásával teszteltük. A legkisebb feladatban ("A" feladat) a műveleti egységek száma 26. Mivel ennek a feladatnak a megoldása a szekvenciális ABB algoritmus segítségével több megoldás keresése esetén is mindössze egy másodperc, ezért ennél kisebb példák vizsgálata párhuzamos feldolgozású algoritmussal már értelmetlen.

A másik tesztelésbe bevont feladat ("B" feladat) 41 műveleti egységet tartalmaz. Ennél a feladatnál már jól látható a végrehajtási idő százalékos csökkenése. A harmadik feladat ("C" feladat) műveleti egységeinek száma ugyan csak 30, de a maximális struktúra bonyolultsága miatt (sok az él a P-gráfban) ennek a feladatnak a megoldása lényegesen hosszabb ideig tart, itt már jelentős gyorsítást jelent abszolút értékben is a párhuzamos feldolgozás.

A tesztelés során a feladatok MILP modelljét oldottuk meg, az időeredmények párhuzamos feldolgozás esetén T800-as transzputeren mért futási eredmények, a szekvenciális futtatások időeredményei a T800-as transzputer sebességének megfelelő Pentium 90-es processzorral ellátott PC-vel mért eredmények.

A tesztelés során az algoritmusok a legjobb 5 megoldást keresték, a teszteredményeket a 5. táblázat tartalmazza.

5. táblázat. Futási eredmények a legjobb 5 megoldás keresése esetén.

processzortopológia	N/A	Gyűrű					
processzorok száma	1	4	8	16	4	8	16
	idő	végrehajtási idő			relatív gyorsítás*		
"A" feladat	0.93	0.71	0.61	0.62	3.05	5.24	10.66
"B" feladat	6.92	3.82	2.15	2.15	2.21	2.49	4.97
"C" feladat	474	160.2	70.3	34.3	1.35	1.19	1.16

* (végrehajtási idő)·(processzorok száma)/(szekvenciális algoritmus végrehajtási ideje)

processzortopológia	N/A	bináris fa					
processzorok száma	1	4	8	16	4	8	16
	idő	végrehajtási idő			relatív gyorsítás*		
"A" feladat	0.93	0.71	0.61	0.62	3.05	5.24	10.66
"B" feladat	6.92	3.81	2.13	2.14	2.2	2.46	4.95
"C" feladat	474	160.2	70.3	34.2	1.35	1.19	1.15

* (végrehajtási idő)·(processzorok száma)/(szekvenciális algoritmus végrehajtási ideje)

A táblázat adatai alapján nyilvánvaló, hogy

(i) nincs lényeges különbség a megvalósított két processzortopológia teljesítménye között. A jobbnak tűnő bináris fa topológia előnye ("közelebb vannak a szolgák a mesterhez") várhatóan csak több processzor használatakor mutatkozik meg határozottabban, illetve olyan feladatok esetén, ahol sok részproblémát kell megoldani, tehát ebből a szempontból nagy a feladat, ugyanakkor egyes részproblémák megoldása kevés időt vesz igénybe, azaz nincs több nagyságrendnyi különbség a kommunikációra fordított időhöz képest.

(ii) Az "A" feladatot nem érdemes párhuzamos feldolgozású algoritmussal megoldani, a kapott gyorsítás sem százalékosan, sem abszolút értékben nem jelentős.

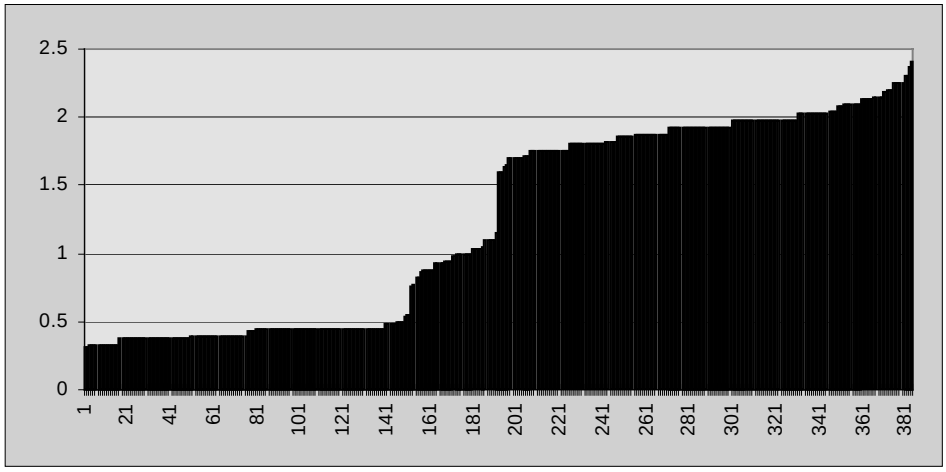
(iii) A "B" feladat már nem tud 16 processzornak feladatot adni, a plusz 8 processzor lényegében csak plusz adminisztrációt jelent erre a feladatra.

(iv) A "C" feladat esetén a párhuzamos algoritmus gyorsulása közel lineáris (a relatív gyorsítás 1-hez közel van). Ez azt mutatja, hogy a kommunikációra

fordított idő elhanyagolható a részproblémák megoldására fordított időhöz képest, továbbá a szolga processzorok terhelése egyenletes.

A processzorok leterheltségét az "A" és "B" feladatok esetén nem érdemes vizsgálni, mindkét feladat esetében maximum 8 processzort érdemes alkalmazni, további futási idő csökkentés nem érhető el (mindkét feladatnál az első részfeladat megoldása után 7 újabb részprobléma keletkezik és ennél soha nem lesz több, azaz 7 szolgánál több alkalmazása felesleges). Az "A" feladatnál minden szolga 1 vagy 2 részfeladatot old meg, míg a "B" feladat esetén 3 vagy 4 részfeladatot.

A "C" feladat megoldását érdemes részletesebben megvizsgálni. Itt a közel lineáris gyorsítás azt mutatja, hogy az egyes feladatok megoldása nagyságrendekkel nagyobb időt vesz igénybe, mint a részfeladathoz kapcsolható kommunikációs idők - feladat kiküldése, eredmény(ek) visszaküldése. Ennél a feladatnál az egyes szolga processzoroknak már jóval több részproblémát kell megoldaniuk (megközelítően: 3 szolga esetén 127, 7 szolga esetén 54, 15 szolga esetén 25 részproblémát). Egyes szolgák által megoldott részproblémák száma természetesen lényegesen eltérhet, mivel a részproblémák megoldására fordított idők is lényegesen eltérhetnek. Ezek az eltérések akkor okoznak egyenetlen leterheltséget, ha több olyan részprobléma van, amely megoldására fordított idő nem tér el lényegesen a hozzá kapcsolódó kommunikációra fordított időtől, ekkor ugyanis az ilyen részfeladatokat megoldó szolgák leterheltsége lényegesen rosszabb lesz, illetve ha az adott szolga a mesterhez "közel" áll a processzortopológiában, a többi processzor maradhat munka nélkül. A "C" feladat részproblémáinak megoldásához szükséges időket a 24. ábra illusztrálja A közel 400 részproblémát a megoldásukhoz szükséges idő (közbülső részprobléma esetén ez természetesen a korlátozó függvény végrehajtásának és a leszármazottak generálásának ideje) szerint rendeztük, így az ábráról jól leolvasható, hogy (i) az egy részprobléma megoldására fordított idő minden esetben több tizedmásodperc, (ii) a részproblémák felénél az egy másodpercet is meghaladja, és (iii) a részproblémák megoldására fordított idők eloszlása egyenletlen.



24. ábra. A "C" feladat részproblémáinak megoldásához szükséges idők (másodpercben).

Az ábrában megadott részproblémák a szekvenciális algoritmus által megoldott részproblémák. A párhuzamos algoritmus néhány plusz részproblémától eltekintve lényegében ugyanezeket oldja meg, ezért a párhuzamos algoritmus által megoldott részproblémákat tartalmazó ábra eltérése a fenti ábrától elhanyagolható lenne (ugyanakkor a végrehajtási idő mérése és különösen az időeredmények továbbítása befolyásolhatja az algoritmus működését, míg a szekvenciális algoritmust csak lassíthatja, de a működését nem befolyásolja).

Ezeket az adatokat az egy részproblémához kapcsolódó kommunikáció idejével kell összehasonlítani. Ehhez megadunk egy durva felső korlátot az egy részproblémához kapcsolódó kommunikáció idejére, ami két részből áll: a mester részfeladatot küld a szolgának, illetve a szolga eredményt (új megoldások, új részproblémák) küld vissza a mesternek. A részfeladat küldése csatorna inicializálásból (szükséges idő 15.5 μsec) és 10 egész szám (40 byte) küldéséből (40×0.58 μsec = 23.2 μsec) áll, azaz a szükséges idő kevesebb, mint 40 μsec. Az eredmény visszaküldése hasonlóan csatorna inicializálásból és maximum 330 valós szám (2640 byte) küldéséből áll (ez utóbbi nagyon extrém eset, 10 megoldás visszaküldését jelenti, ami nagyon durva felső korlát). Egy részproblémához kapcsolódó kommunikáció ideje tehát kevesebb, mint 1600 μsec = 0.0016 sec.

Mint látható, a részproblémák megoldásához szükséges idők közül a legkisebb is nagyságrendekkel nagyobb a kommunikációra fordított időnél. Azaz a mester

a 16 szolgánál jóval többet tudna irányítani úgy, hogy azok terhelése egyenletes maradna.

A teszteredmények azt mutatják, hogy a párhuzamos feldolgozású gyorsított branch-and-bound algoritmus alkalmazása indokolt bonyolult feladatok - amely nem feltétlenül jelent nagy feladatot - megoldása esetén.

11. Összefoglalás

A dolgozatban bemutattuk a következő folyamat-hálózatszintézis feladat kiterjesztéseket: hulladékkezeléssel integrált folyamat-hálózatszintézis, integrált folyamat-hálózat- és irányítórendszer tervezés, folyamat-hálózatszintézis szakaszosan folytonos célfüggvényű műveleti egységekkel, folyamat-hálózatszintézis feladat megoldása párhuzamos működési elvű számítógépeken.

Mindegyik kiterjesztés megoldására megadtunk egy kombinatorikus jellegű, a feladattípus jellegzetességeit jól kihasználó megoldó módszert. A módszerek mindegyike lehetővé teszi az optimális megoldások hatékony keresését, matematikailag jól megalapozott feladat reprezentációra építenek, az optimális megoldás megtalálása mindegyik módszer esetében garantált.

Mind az itt ismertetett, mind más kiterjesztések kombinált alkalmazása lehetséges, ezzel gyakorlati szempontból fontos, ugyanakkor hagyományos módszerekkel nehezen vagy nem megoldható feladatosztályok megoldása válik lehetségessé.

12. Irodalomjegyzék

[1] Berger, S. A., The Pollution Prevention Hierarchy as an R&D Management Tool, *AIChE Symposium Series*, Volume on Pollution Prevention via Process and Product Modifications (Eds: M. M. El-Halwagi and D. P. Petrides), **90**(303), 23-29 (1995).

[2] Brendel, M. H., F. Friedler, and L. T. Fan, Combinatorial Foundation for Logical Formulation in Process Network Synthesis, accepted for publication in *Computers chem. Engng* (1998).

[3] Constantinou, L., C. Jakslund, K. Bagherpour, R. Gani, and I. D. L. Bogle, Application of the Group Contribution Approach to Tackle Environmentally Related Problems, *AIChE Symposium Series*, Volume on Pollution Prevention via Process and Product Modifications (Eds: M. M. El-Halwagi and D. P. Petrides), **90**(303), 105-117 (1995).

[4] Crabtree, E. W. and M. M. El-Halwagi, Synthesis of Environmentally Acceptable Reactions, *AIChE Symposium Series*, Volume on Pollution Prevention via Process and Product Modifications (Eds: M. M. El-Halwagi and D. P. Petrides), **90**(303), 117-128 (1995).

[5] Duran, M. A. and I. E. Grossmann, An Outer-Approximation Algorithm for a Class of Mixed-integer Nonlinear Programs, *Math. Programming*, **36**, 307 (1986).

[6] Eckstein, J., Parallel branch-and-bound algorithms for general mixed integer programming on the CM-5, Technical report TMC-257, Thinking Machines Corporation, Cambridge, MA. U.S.A. (1993).

[7] Floudas, C. A. and V. Visweswaran, A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLPs - I. Theory, *Computers chem. Engng*, **14**, 1397-1417 (1990).

[8] Fraga, E. S. and K. I. M. McKinnon, The Use of Dynamic Programming with Parallel Computers for Process Synthesis, *Computers chem. Engng*, **18**, 1-13 (1994).

[9] Fraga, E. S. and K. I. M. McKinnon, Portable Code for Process Synthesis Using Workstation Clusters and Distributed Memory Multicomputers, *Computers chem. Engng*, **19**, 759-774 (1995).

[10] Friedler, F., K. Tarjan, Y. W. Huang, and L. T. Fan, Combinatorial Algorithms for Process Synthesis, *Computers chem. Engng*, **16**, S313-20 (1992).

[11] Friedler, F., K. Tarjan, Y. W. Huang, and L. T. Fan, Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems, *Chem. Eng. Sci.*, **47**(8), 1973-1988 (1992).

[12] Friedler, F., K. Tarjan, Y. W. Huang, and L. T. Fan, Graph-Theoretic Approach to Process Synthesis: Polynomial Algorithm for Maximal Structure Generation, *Computers chem. Engng*, **17**(9), 929-942 (1993).

[13] Friedler, F., J. Varga, and L. T. Fan, A Combinatorial Approach to the Multiperiod Synthesis of the Structure of Process Industries, *Proceedings of the ESCAPE-3* (European Symposium on Computer Aided Process Engineering), 2-6 (1993).

[14] Friedler, F., J. B. Varga, and L. T. Fan, Decision-Mapping: A Tool for Consistent and Complete Decisions in Process Synthesis, *Chem. Engng Sci.*, **50**(11), 1755-1768 (1995).

[15] Friedler, F., J. B. Varga, and L. T. Fan, Algorithmic Approach to the Integration of Total Flowsheet Synthesis and Waste Minimization, *AIChE Symposium Series*, Volume on Pollution Prevention via Process and Product Modifications (Eds: M. M. El-Halwagi and D. P. Petrides), **90**(303), 86-97 (1995).

[16] Friedler, F., J. B. Varga, E. Feher, and L. T. Fan, Combinatorially Accelerated Branch-and-Bound Method for Solving the MIP Model of Process Network Synthesis, *Nonconvex Optimization and Its Applications, State of the Art in Global Optimization, Computational Methods and Applications* (Eds: C. A. Floudas and P. M. Pardalos), 609-626, Kluwer Academic Publishers, Norwell, MA, U.S.A. (1996).

[17] Friedler, F., L. T. Fan, and B. Imreh, Process Network Synthesis: Problem Definition, *Networks*, **28**(2), 119-124 (1998).

[18] Gál, I. P., Gráfelméleti módszerek alkalmazása az irányításelméletben, Ph.D. értekezés (1997).

[19] Gál, I. P., J. B. Varga, and K. M. Hangos, Integrated Structure Design of a Process and Its Control System, *J. Proc. Cont.*, **8**(4), 251-263 (1998).

[20] Georgiou, A., and C. A. Floudas, Simultaneous Process Synthesis and Control: Minimization of Disturbance Propagation in Heat Recovery Systems. *Foundations of Computer-Aided Process Design* (Eds: J. Siirola, J., I. E. Grossmann, and G. Stephanopoulos), 435-450 (1990).

[21] Hangos, K. M., F. Friedler, J. Varga, and L. T. Fan, Graph-Theoretic Approach to Integrated Process and Control System Synthesis, *Proceedings of the IFAC '94* (International Federation of Automatic Control), Workshop on Integration of Process Design and Control, 58-63 (1994).

[22] Hangos, K. M., J. B. Varga, F. Friedler, and L. T. Fan, Integrated Synthesis of a Process and its Fault-Tolerant Control System, *Computers chem. Engng*, **19**, S465-470 (1995).

[23] High, K. A. and R. D. LaRoche, Parallel Nonlinear Optimization Techniques for Chemical Process Design Problems, *Computers chem. Engng*, **19**, 807-827 (1995).

[24] Ibaraki, T., Theoretical Comparisons of Search Strategies in Branch-and-Bound Algorithms, *Journal of Computer and Information Science*, **5**, 315-344 (1976).

[25] Kocis, G. R., and I. E. Grossmann, A Modeling and Decomposition Strategy for the MINLP Optimization of Process Flowsheets, *Computers chem. Engng*, **13**, 797-819 (1989)

[26] Kudva, G. K. and J. F. Pekny, DCABB a Distributed Control Architecture for Branch and Bound Calculations, *Computers chem. Engng*, **19**, 847-866 (1995).

[27] Lawler, E. L. and D. E. Wood, Branch-and-Bound Methods: A Survey, *Operations Research*, **14**(4), 699-719 (1966).

[28] Luyben, M. L. and C. A. Floudas, A Multiobjective Optimization Approach for Analysing the Interaction of Design and Control, Part 1: Theoretical Framework, *Interactions between Process Design and Process Control* (Ed. J. D. Perkins), 101-106, Pergamon Press, Oxford (1992).

[29] Meyer, R. R. and S. A. Zenios, Parallel Optimization on Novel Computer Architectures, *Ann. Opn. Res.*, **14** (1988).

[30] Nichida, N. and A. Ichikawa, Synthesis of Optimal Dynamic Process Systems by a Gradient Method, *Ind. Eng. Chem. Proc. Des. Dev.*, **14**(3), 236-242 (1975).

[31] Nichida, N., Y. A. Liu, and A. Ichikawa, Studies in Chemical Process Design and Synthesis, part 2: Optimal Synthesis of Dynamic Process Systems with Uncertainty, *AIChE Journal*, **22**(3), 539-549 (1976).

[32] Pardalos, P. M., A.T. Philips, and J. B. Rosen, Topics in Parallel Computing in Mathematical Programming, Science Press, New York (1992).

[33] Quinn, M. J., Designing Efficient Algorithms for Parallel Computers, McGraw-Hill Computer Science Series, New York (1987).

[34] Raman, R. and I. E. Grossmann, Symbolic Integration of Logic in MILP Branch and Bound Methods for the Synthesis of Process Networks, *Annals of Operations Research*, **42**, 169-191 (1993).

[35] Rittmeyer, R. W., Prepare an Effective Pollution-Prevention Program, *Chemical Engineering Progress*, **87**, 56-62 (1991).

[36] Schnabel, R. B., Concurrent Function Evaluations in Local and Global Optimization, *Comp. Meth. Appl. Mech. Engng*, **64**, 537-552 (1987).

[37] Varga, E. I. and K. M. Hangos, The Effect of Heat Exchanger Network Topology on the Network Control Properties, *Control Engineering Practice*, **1**, 375-380 (1993).

[38] Varga, J. B., F. Friedler, and L. T. Fan, Parallelization of the Accelerated Branch and Bound Algorithm of Process Synthesis: Application in Total Flowsheet Synthesis, *Acta Chimica Slovenica*, **42**(1), 15-20 (1995).

[39] Varga, J. B., F. Friedler, and L. T. Fan, Risk Reduction through Waste Minimizing Process Synthesis, *Proceedings of the 21st Annual RREL* (Risk Reduction Engineering Laboratory) Research Symposium, 359-363 (1995).

[40] Visweswaran, V. and C. A. Floudas, A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLPs - II. Application of Theory and Test Problems, *Computers chem. Engng*, **14**, 1419-1434 (1990).

13. Függelék

13.1. Branch-and-bound

A “branch-and-bound” egy eszköztípus összefoglaló neve, jól ismert technika optimalizálási feladatok megoldására [27], sok változata ismert [24], itt csak egy olyan általános formális leírást adunk meg, ami szükséges a folyamat-hálózatszintézis megoldására kidolgozott gyorsított branch-and-bound algoritmus definiálásához.

A feladat definiálásához meg kell adnunk a lehetséges megoldások halmazát (feltételrendszer), valamint azt a szempontot (célfüggvény), ami szerint az optimális megoldást keressük:

Tegyük fel, hogy adott egy f célfüggvény, amelyet a lehetséges megoldások S_0 halmazán kell minimalizálnunk. Bevezetünk egy a lehetséges megoldások halmazánál nem szűkebb P_0 halmazt, e halmaz részhalmazait nevezzük részproblémáknak. A P_i részprobléma optimális megoldásainak halmazát $Opt(P_i)$-vel, az optimum értékét $F(P_i)$-vel jelöljük. Ha $Opt(P_i)$ halmaz üres, akkor $F(P_i)$ értékét végtelennek tekintjük.

A szétválasztó lépésben egy részproblémát bontunk részhalmazaira, új részproblémákat generálva. A szétválasztó lépés után egy P_i részproblémából generált új részproblémák halmazát $son(P_i)$-vel jelöljük. A son operátornak az alábbi tulajdonságokkal kell rendelkeznie: (i) a $son(P_i)$ elemeinek uniója P_i-t adja, (ii) a $son(P_i)$ minden eleme valódi részhalmaza P_i-nek. Ha a P_i halmaz üres, akkor $Opt(P_i)$ is az, így P_i-re a szétválasztás nem folytatódik. Nem szükséges feltétel, de a hatékonyságot javítja, ha a $son(P_i)$ elemei diszjunkt halmazok.

Legyen a G az F olyan korlátozó függvénye, amely a következő feltételeket teljesíti:

(i) $G(P_i) \leq F(P_i)$ minden $P_i \in \wp(P_0) \setminus \{\varnothing\}$ részproblémára,

(ii) $G(P_i) = \infty$ ha $P_i = \emptyset$ és

(iii) $G(P_{ij}) \geq G(P_i)$, $P_{ij} \in son(P_i)$.

A megoldandó részproblémák közül bármelyik választható további feldolgozásra, melyet az s keresőfüggvény határoz meg. Összefoglalva, a branch-and-bound algoritmus pontos megadásához szükség van egy szétválasztó (son), egy kereső (s) és egy korlátozó (G) függvényre.

Az algoritmus végességének biztosításához azonban vagy további feltételeket kell a korlátozó függvénynek teljesítenie, vagy alsó és felső korlátokat alkalmazva adhatunk lezárási feltétel(eke)t, melyek teljesülése esetén az adott részprobléma dekomponálása nem folytatódik. Ilyen feltétel lehet például a vegyes egész programozási feladatoknál az egész változók értékének egyértelműsítése után a $G(P_i) = F(P_i)$ feltétel. Ekkor a korlátozó függvény minimumhelye lehetséges megoldás lesz. Folyamat-hálózatszintézis feladat megoldásakor ez a feltétel a megoldás műveleti egységeinek pontos megadása után az eredeti költségfüggvény alkalmazását jelenti. Megfelelő megállási feltétel teljesülése után rendelkezésünkre áll az $Opt(P_i)$ halmaz, illetve az $Opt(P_i)$ egy részhalmaza, ha a korlátozó függvény egy optimális megoldást ad meg valamely részfeladatra optimális megoldáshalmaz esetén is.

A branch-and-bound módszer lépései a következők.

1. *Inicializálás*

   $A := \{P_0\}$; $z := \infty$; $sol := \emptyset$;

2. *Keresés*

   Ha $A = \emptyset$, ugrás a 7. lépésre, egyébként $P_i := s(A)$;

3. *Teszt*

   Ha $G(P_i) = \infty$ vagy $G(P_i) > z$, akkor ugrás a 6. lépésre;

   Ha a megállási feltétel teljesül, ugrás az 5. lépésre;

4. *Szétválasztás*

   $A := (A \cup son(P_i)) \setminus \{P_i\}$; ugrás a 2. lépésre;

5. *Frissítés*

Ha $z>F(P_i)$, sol := $Opt(P_i)$;

Ha $z = F(P_i)$, sol := sol$\cup Opt(P_i)$;

z := min(z, $F(P_i)$);

(Közbülső részprobléma esetén az $Opt(P_i)$ és az $F(P_i)$ helyett alkalmazható a G függvény által meghatározott optimális megoldás illetve az F függvény értéke erre a megoldásra.)

6. *Lezárás*

   A := A\{P_i}; ugrás a 2. lépésre;

7. *Megállás*

   Ekkor sol $\subseteq Opt(S_0)$ és $z = F(S_0)$.

A branch-and-bound algoritmus garantálja optimális megoldás és az optimum értékének megtalálását, ha az létezik. Ha az optimum értéke $\propto$ az algoritmus lefutása után, akkor nincs lehetséges megoldás. A branch-and-bound típusú algoritmusok további előnyei: az optimális megoldás mellett az n legjobb megoldás generálásának lehetősége, a heurisztikus algoritmusokkal való kombinálhatóság, valamint a jó megvalósíthatóság párhuzamos működési elvű számítógépeken [6, 33].