

TDK-dolgozat

Szili László



Pannon Egyetem



Szétválasztási hálózatok alternatív optimális struktúráinak generálása modelltranszformációval

Készítette: Szili László

II. évfolyam

Témavezető: Dr. Bertók Botond

egyetemi docens

Műszaki Informatikai Kar, Rendszer- és Számítástudományi Tanszék

Tartalomjegyzék

1. Bevezetés	4
2. Reakcióút meghatározás	6
2.1. Reakcióút-szintézis feladat	6
2.2. Lehetséges reakcióutak	6
2.3. PNS és RPI kapcsolata	7
2.4. P-gráf reprezentáció	7
2.5. Kombinatorikusan lehetséges reakcióutak	9
2.6. Algoritmusok	10
2.6.1. RPIMSG	10
2.6.2. RPISSG	10
2.6.3. RPIPBT	11
3. Szétválasztási hálózat szintézis	13
3.1. Irodalmi áttekintés	13
3.2. Alapfogalmak	13
3.3. Szigorú szuperstruktúra, és redukált szigorú szuperstruktúra	14
4. SNS-RPI modelltranszformáció	21
4.1. A két feladat ekvivalenciája	21
4.1.1. P-gráf reprezentáció az SNS feladathoz	22
4.2. Algoritmus	24
4.3. További tulajdonságok	27
4.4. Megvalósítás	28
5. Megoldott feladatok	30
5.1. 1. Feladat	30
5.2. 2. Feladat	31
5.3. 3. Feladat	31
6. Összefoglalás	33
7. Függelék	36
7.1. A programhoz tartozó bemeneti, és generált modell	36
7.1.1. Az 1.feladat bemenete	36
7.1.2. Az 1.feladat kimenete	36
7.1.3. Az 2.feladat bemenete	37

7.1.4.	Az 2.feladat kimenete	38
7.1.5.	Az 3.feladat bemenete	39
7.1.6.	Az 3.feladat kimenete	40

Absztrakt

A szétválasztási hálózatok fontos szerepet töltenek be az ipar számos területén. Jelen vannak a vegyipartól kezdve, az élelmiszeriparon és a mezőgazdaság feldolgozó iparain keresztül egészen a gyógyszeriparig, legfontosabb alkalmazásuk pedig az olajiparban van a mai napig. Egy ilyen hálózat legfontosabb elemei a szétválasztók, de tartalmazhat megosztókat, keverőket és csővezetékeket is. A feladat ezekből az ismert eszközökből, a kívánt szétválasztást megvalósító optimális hálózat tervezése. Jelentőségét az adja, hogy egy ilyen hálózat költsége egy termelőrendszer költségének jelentős részét teszi ki. Egy adott feladatot számos eltérő topológiájú hálózat is meg tud valósítani, ezek közül valamilyen szempontból legkedvezőbb, általában legolcsóbb, megoldást szeretnénk megtalálni. Bizonyos esetekben előfordulhat, hogy a kapott optimális megoldást nem célszerű, vagy nem is lehetséges megvalósítani a termelőrendszer fizikai adottságai miatt, így szükségünk lehet olyan megoldásokra, melyek minimálisan térnek el az optimálistól, de szerkezetileg eltérően épülnek fel. A szakirodalomban nem publikált olyan módszer, amely egy adott problémára nem csak az optimális megoldást, hanem különböző alternatívákat is generál. Jelen dolgozatomban ezt a problémát oldottam meg az szétválasztó hálózat szintézis (SNS) feladat reakció út meghatározás (RPI) feladatosztályba való átfogalmazásával, valamint beláttam a két feladat ekvivalenciáját, módszert adva alternatívák generálására egy adott SNS feladathoz.

1. fejezet

Bevezetés

A szétválasztási hálózatok széles körben fontos szerepet töltenek be az iparban. Jelen vannak a vegyipartól kezdve, az élelmiszeriparon keresztül egészen a gyógyszeriparig, legfontosabb alkalmazásuk pedig az olajiparban van. Egy ilyen hálózat legfontosabb elemei a szétválasztók, de tartalmazhat megosztókat, keverőket és csővezetékeket is. Jelentőségük abból származik, hogy egy gyártórendszer megvalósítási költségének nagyrészt egy ilyen hálózat költsége teszi ki.

Dolgozatom célja egy olyan módszer kidolgozása volt, mely képes alternatív optimális megoldások generálására egy olyan szétválasztási hálózat szintézis problémára, amely egyszerű, éles szétválasztókat tartalmaz. Erre azért van szükség, mert egy optimális topológia nem minden esetben valósítható meg egy gyártórendszer fizikai adottságai miatt.

Mielőtt kitérnék a saját eredményeim részletezésére, dolgozatomban bemutatom mind a reakcióút meghatározás (angol elnevezéssel reaction pathway identification (RPI)), mind pedig a szétválasztási hálózat szintézis (angol elnevezéssel separation network synthesis (SNS)) feladat felépítését, tulajdonságait és megoldási módszerét. A 2. fejezetben részletesen az RPI feladattal foglalkozom. Az általános feladatleírason és tulajdonságai bemutatásán túl ismertetem, hogy milyen algoritmusok léteznek a feladat megoldására, és azok mi alapján működnek. Így egy alapot képezve ahhoz, hogy a későbbiekben ezt alkalmazni tudjuk a modelltranszformációt követően.

A következő fejezetben az SNS feladat bemutatása mellett fő szerepet szántam a szigorú szuperstruktúra fogalmának bevezetésére, generálására, és a struktúra egyszerűsítésére. Mindezt azon megfontolás alapján, mivel egy SNS probléma megoldásának optimalitása, és helyállósága nagymértékben függ a használt struktúra felépítésétől, és helyességétől. Ezek után áttérek a saját eredményeimre, ahol bemutatom a kidolgozott eljárást, valamint belátom a két modell ekvivalenciáját is. Az alternatívák generálását oly módon oldottam meg, hogy először az SNS feladatból felépítettem egy RPI modellt, majd a kapott struktúrát az RPI feladat megoldására alkotott algoritmussal oldottam meg. A módszer hatékonyságát mutatja az 5. fejezetben a szakirodalomból vett példák megoldása, ahol minden esetben legalább a publikált optimális, vagy annál jobb megoldást adtam meg a számos optimálishoz közeli, de eltérő felépítésű megoldások mellett.

Folyamathálózat-szintézis komponens megmaradással

A folyamathálózat-szintézis feladat olyan folyamatok meghatározásával foglalkozik, melyek adott bemeneti anyagokból adott lépéseken keresztül, előre meghatározott anyagokat állít elő. Minden lépés be és kimenete anyag természetű. Egy anyagalmazból másik előállítását végző lépéseket műveleti egységeknek hívjuk. A vizsgált esetben véges sok különböző anyag adható meg, melyek a hálózat, vagy egyes műveleti egységek ki és bemeneteit azonosítják egyben. Két műveleti egység be és kimenete akkor kapcsolható össze, ha ugyanaz az anyag van mindkettőhöz rendelve.

A folyamathálózat-szintézis feladatok egy olyan osztályát vizsgáljuk, ahol az anyagok véges sok összetevőből állhatnak előre adott arányban, és az összetevőikre megmaradás teljesül. Valamint minden műveleti egységre igaz, hogy kimenő anyagai minden összetevőből pontosan annyit tartalmaznak, mint a bemenő anyagai, és minden folyamatra igaz, hogy végtermékei minden összetevőből annyit tartalmaznak, mint nyersanyagai.

A feladatot egy $(\mathbf{E}, O, \mathcal{M}, Q)$ négyessel adhatjuk meg. $Q = \{q_1, q_2, \dots, q_h\}$ az anyagok véges sok összetevőjének rendezett halmaza. $\mathcal{M} = \{a_1, a_2, \dots, a_l\}$ az anyagok véges rendezett halmaza, melyben minden anyagot egy olyan $\mathbf{a}_j = (\mathbf{a}_{1,j}, \mathbf{a}_{2,j}, \dots, \mathbf{a}_{h,j}) \in (R_0^+)^h$ nemnegatív számokból álló vektor ír le, ahol $a_{k,j}$ a q_k összetevő mennyisége az $\mathbf{a}_j$ anyagban ($k = 1, 2, \dots, h$). A szintetizálendő eredő folyamatot az $\mathbf{E} = (E_1, E_2, \dots, E_l) \in \mathbf{R}^l$ valós számokból álló l dimenziós vektor adja meg, ahol E_j a folyamat által az a_j anyagból előállított és felhasznált mennyiség különbsége ($j = 1, 2, \dots, l$). $\mathbf{a}_j$ a folyamat nyersanyaga pontosan akkor, ha $E_j < 0$, és $\mathbf{a}_j$ a folyamat végterméke pontosan akkor, ha $E_j > 0$. Végül $O = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n\}$ a műveleti egységek véges rendezett halmaza, melyben minden $\mathbf{e}_i$ műveleti egységet egy $\mathbf{e}_i = (e_{1,i}, e_{2,i}, \dots, e_{l,i}) \in \mathbf{R}^l$ valós számokból álló l dimenziós vektor azonosít ahol $e_{j,i}$ az i -edik műveleti egység által előállított és felhasznált mennyiségek különbsége ($j = 1, 2, \dots, l$).

2. fejezet

Reakcióút meghatározás

2.1. Reakcióút-szintézis feladat

Reakcióutak meghatározása szükséges, ha kémiai, vagy biokémiai reakciók mechanizmusát kívánjuk megismerni. A reakcióút elemi reakciók lépéseiből áll, melyek a reagens anyagoktól elvezetnek a végtermékekig. A feladatban a meghatározandó reakcióutat az eredő reakció írja le. Az eredő reakció reakcióútjának meghatározása két részből áll. Az első szakasz az összes lehetséges mechanizmust azonosítja, majd a második szakaszban kiválasztásra kerül a pontos reakcióút az első szakaszban azonosítottak közül. Szemléltető példaként a bután (C_4H_{10}) buténné (C_4H_8) való dehidrogénezése feladatot mutatom be, mely folyamat eredő reakcióját, és elemi reakcióit megtalálhatjuk az 2.1 ábrán.

A módszer amit alkalmazunk, a folyamathálózat-szintézis számára megfogalmazott axiomatikus módszerben gyökerezik. Feltesszük, hogy a hálózatban szereplő részecskék koncentrációja állandó, nem oszcillál és nem tranziens jellegű. Továbbá, hogy az eredő reakció, és az elemi reakciók halmaza előre adott. Mivel minden elemi reakció megfordítható, így elégséges a kiindulási anyagoktól a végtermékekig vezető reakcióutakat az eredő reakció egyik irányának megfelelően meghatározni. További szükséges feltételként megemlítendő, hogy eredő reakció, és az elemi reakciók teljesítik az anyagmegmaradást, valamint az hogy a mikroszkopikus megfordíthatóság elve alapján egy reakcióút sem tartalmazhat nulla eredőjű köröket. Nulla eredőjű kört például a 2.1. ábrán szereplő $(1 \rightarrow), (3 \leftarrow)$ és $(5 \leftarrow)$ elemi reakciók alkotnak, mivel minden anyagból ugyanannyi keletkezik, mint amennyi felhasználásra került. A sorszám mellett a nyilak azt jelentik, hogy az adott elemi reakció mely irányát tekintjük.

Egy reakcióút-szintézis feladatot egy $(\mathbf{E}, O, \mathcal{M})$ hármas definiál, ahol $\mathbf{E}$ az eredő reakció, $O = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n\}$ az elemi reakciólépések véges rendezett halmaza, $\mathcal{M} = \{a_1, a_2, \dots, a_l\}$ pedig a kémiai vagy aktív átmeneti részecskék véges rendezett halmaza. $\mathbf{E} = \{E_1, E_2, \dots, E_l\} \in \mathbf{R}^l$ egy valós számokból álló l dimenziós vektor, ahol E_j az eredő reakció által az a_j kémiai részecskéből előállított és felhasznált mólok számának különbsége ($1 \leq j \leq l$). Egy elemi reakciólépést is egy $\mathbf{e}_i = (e_{1,i}, e_{2,i}, \dots, e_{l,i}) \in \mathbf{R}^l$ valós számokból álló l dimenziós vektor azonosít, ahol $e_{j,i}$ az i -edik elemi reakciólépés által az a_j kémiai vagy aktív részecskéből előállított és felhasznált mólok számának különbsége ($1 \leq j \leq l$).

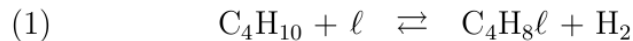
2.2. Lehetséges reakcióutak

A korábban felsorolt alapelvek, és tulajdonságok alapján megadhatjuk a lehetséges reakcióutak halmazát definiáló axiómahalmazt. Lehetséges reakcióúton olyan reakcióutat értünk, ami teljesíti alapvető elvárásainkat, és az adott bemeneti anyagokból kiindulva, csak adott elemi reakciókon keresztül képes szolgáltatni a kívánt

Eredő reakció:



Elemi reakciók:



2.1. ábra. A bután buténné történő dehidrogénezése

végtermékeket. Az elemi reakciók egy halmazát egy feladatra nézve lehetséges reakcióútnak nevezzük, ha teljesíti a következő axiómákat:

(R1) Minden végtermék az eredő reakcióban szereplő mértékben kerül előállításra.

(R2) Minden kiindulási reagens az eredő reakcióban szereplő mértékben kerül felhasználásra.

(R3) A reakcióútban szereplő valamely elemi lépés által előállított minden átmeneti aktív részecskét a reakcióútban szereplő más lépések teljes mértékben felhasználnak, illetve a reakcióútban szereplő valamely elemi lépés által felhasznált minden átmeneti aktív részecskét a reakcióútban szereplő más lépések teljes mértékben előállítják.

(R4) A reakcióútban szereplő minden elemi lépés a feladatban meghatározott.

(R5) A reakcióutat reprezentáló hálózat nem tartalmaz nulla eredőjű kört.

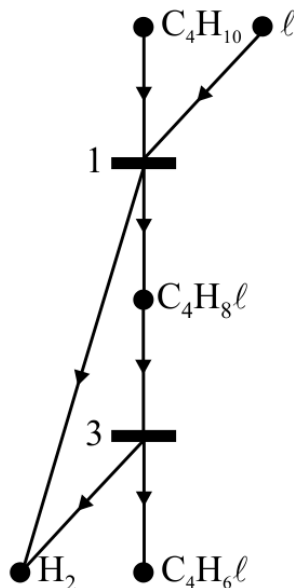
A reakcióút-szintézis feladat kombinatorikus jellege nagyon erős, ezért meg kívánjuk a későbbiekben határozni a lehetséges reakcióutak strukturális jellemzőit is.

2.3. PNS és RPI kapcsolata

Egy reakcióút-szintézis feladat megfelel a komponens-megmaradással rendelkező folyamathálózat-szintézis feladatnak, csak szóhasználatban különbözik. Itt az anyagokat kémiai vagy átmeneti részecskének nevezzük, a műveleti egységeket elemi reakciólépéseknek, a meghatározandó folyamatot pedig eredő reakciónak. A részecskéket összegképletük alapján egyértelműen azonosíthatjuk, így egyben összetételüket, és a megmaradásokat is ellenőrizhetjük segítségével. A reakcióút-szintézis feladat azonban annyiban tér el, hogy itt minden elemi reakció fordítottja is szerepelhet.

2.4. P-gráf reprezentáció

Egy egyértelmű halmaz reprezentációra van szükségünk a reakció-út meghatározásnál, ha ábrázolni szeretnénk a hálózatot amit több elemi reakció kombinációjából nyerünk. Az ábrázolásra azért van szükség, hogy a



2.2. ábra. Az $(1 \rightarrow)$ és a $(3 \rightarrow)$ elemi reakciólépést reprezentáló P-gráf

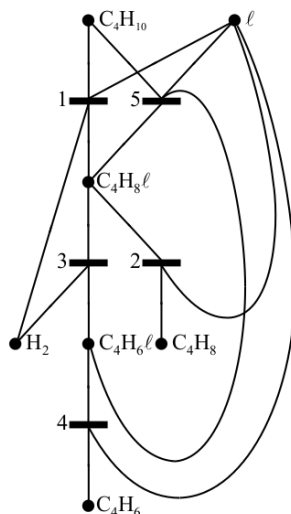
struktúrát a későbbiekben formálisan tudjuk vizsgálni.

Az elemi reakció lépések irányítottak, ezért minden belőlük álló hálózat vázolható irányított gráfokkal. A hagyományos értelemben vett gráfok azonban alkalmatlanok az ilyen hálózatok egyértelmű leírásához, ebből kifolyólag a folyamathálózatok számára bevezetett kombinatorikus modellt, a P-gráfot használjuk. Az elemi reakciólépések egy $o \subseteq O$ halmazát és a kémiai vagy aktív átmeneti részecskék $m \subseteq \mathcal{M}$ halmazát reprezentáló P-gráfot egy (m, o) pár határoz meg, ahol az m halmaz legalább az o halmazban szereplő elemi reakciólépések által előállított vagy felhasznált kémiai vagy aktív részecskéket tartalmazza. A P-gráf egy olyan $(\mathcal{V}, \mathcal{A})$ páros gráf, melynek csúcsai rendre a kémiai vagy aktív részecskéket illetve az elemi reakciólépéseket reprezentálják ($\mathcal{V} = m \cup o$). Az m halmazban szereplő részecskéket azonosító csúcsokat M-típusú, az o halmazban szereplő elemi reakciólépéseket azonosító csúcsokat pedig O-típusú csúcsoknak nevezzük. Továbbá az élek irányítása megfelel az elemi reakciólépés irányának, valamint az elemi lépések által felhasznált részecskéktől vezetnek az elemi lépésig, onnét pedig az általa előállított részecskékig. Egy P-gráf M-típusú csúcsait tele karikákkal, az O-típusú csúcsait pedig vízszintes vonalakkal jelöljük. Ha egy elemi reakció mindkét irányban szerepel, akkor is csak egy csúcs reprezentálja, melyhez kapcsolódó élek nem irányítottak (mindkét irányban irányítottak).

A P-gráf reprezentáció segítségével kimondhatunk két tételt a lehetséges reakcióutakról. A tételek egy (m, o) P-gráfon ábrázolt lehetséges reakcióút két szükséges struktúrális tulajdonságát fogalmazzák meg, valamint előkészítik a kombinatorikusan lehetséges reakcióutak tulajdonságainak kimondását. (a tételek bizonyításai megtalálható itt: [3])

Tétel

Minden lehetséges reakcióutat reprezentáló (m, o) P-gráfban, bármely $a_j \in m$ kémiai vagy átmeneti részecskétől vezet út legalább egy végtermékig.



2.3. ábra. A bután dehidrogénezése feladat ábrázolása P-gráffal

Tétel

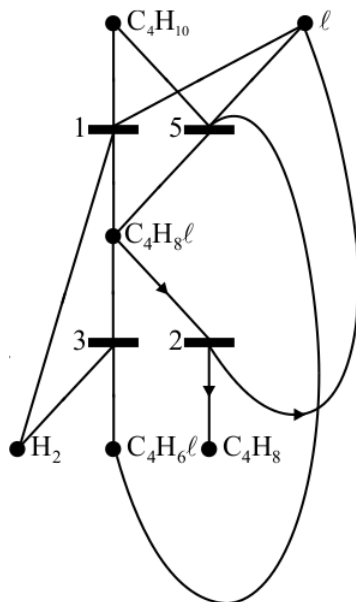
Minden lehetséges reakcióutat reprezentáló (m, o) P-gráfban, bármely $a_j \in m$ kémiai, vagy átmeneti részecskeig vezet út legalább egy kiindulási reagenstől.

2.5. Kombinatorikusan lehetséges reakcióutak

A reakcióutak P-gráf reprezentációja lehetőséget ad a struktúrális tulajdonságok vizsgálatára. Erre azért van szükség, hogy megfogalmazzhassuk, majd formális eszközökkel vizsgálhassuk a lehetséges reakcióutak kombinatorikus tulajdonságait. A lehetséges reakcióutak (R1), (R2), ..., (R5) axiómája újrafogalmazható a következő hét kombinatorikus tulajdonság formájában.

- (T1) Az eredő reakció által előállított minden végtermék szerepel a gráfban.
- (T2) Az eredő reakció által felhasznált minden kiindulási reagens szerepel a gráfban.
- (T3) Minden O-típusú csúcs a reakció-szintézis feladatban definiált elemi reakciólépést reprezentál.
- (T4) Minden kémiai vagy aktív átmeneti részecskét reprezentáló csúcsból vezet a gráfban út legalább egy végtermékig.
- (T5) Ha egy M-típusú csúcs része a gráfnak, akkor vezet hozzá él legalább egy O-típusú csúcsból, vagy vezet belőle él legalább egy O-típusú csúcsba.
- (T6) Ha egy M-típusú csúcsba nem vezet él a gráfban, akkor kiindulási reagenst reprezentál.
- (T7) Egy elemi reakció előre és fordított lépése közül legfeljebb az egyik szerepel a gráfban.

Egy P-gráffal szemléltetett elemi reakciók struktúráját kombinatorikusan lehetséges reakcióútnak nevezzük, ha teljesítik a felsorolt kombinatorikus tulajdonságokat (a tulajdonságok bizonyítása megtalálható itt: [3]).



2.4. ábra. Az RPIMSG algoritmus lebontó fázisa után maradó struktúra a bután dehidrogénezése feladatra

2.6. Algoritmusok

A korábban bevezetett axiómák, kombinatorikus tulajdonságok valamint a P-gráf reprezentáció biztos alapot nyújtottak hatékony algoritmusok megfogalmazására, melyeket ebben az alfejezetben kívánok bemutatni.

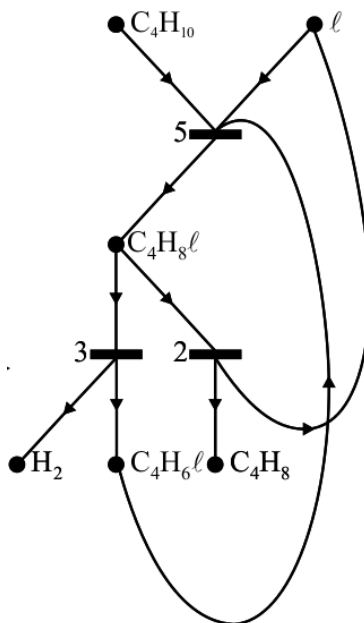
2.6.1. RPIMSG

Egy reakcióút felépítésében minden megadott elemi reakció szerepelhet előre, fordított vagy egyik irányban sem. Így összesen a három lehetőség $3^n - 1$ kombinációját kell figyelembe venni (azt az esetet elhanyagolhatjuk, amikor egy elemi reakció sem szerepel a hálózatban). Mivel a megoldások száma exponenciális függvénye a megadott elemi reakcióknak, így nem lehet olyan algoritmus, ami ezen megoldásokat exponenciálisnál kisebb lépésszámmal leszámolná. Jelentős gyorsulást abban az esetben érhetünk el, ha tudjuk csökkenteni a szükséges elemi reakciók számát úgy, hogy ne veszítsünk egyetlen megoldást sem.

Ezt a feladatot végzi az RPIMSG algoritmus. Az algoritmus két részből áll. Az első része egy lebontó fázis, melyben öt esetet különböztet meg, és valamelyik teljesülését követően eltávolít olyan elemi reakciókat vagy részecskéket, amelyek nem szerepelhetnek egyetlen kombinatorikus lehetséges struktúrában sem, majd rekurzívan hívja önmagát, ameddig lehetséges, hogy a struktúra tartalmaz eltávolítandó elemi reakciókat. A második egy felépítő fázis, melyben a végtermékektől kiindulva az élek irányításával szembe haladva összegyűjti azokat a megmaradt részecskéket, és elemi reakciókat, amelyekből vezet a gráfban út egy végtermékig. Ha befejezte futását visszatér a maximális struktúrával.

2.6.2. RPISSG

Az RPISSG algoritmus feladata az összes kombinatorikusan lehetséges reakcióút generálása. Működése a következőképp történik: a (T4) tulajdonság alapján a végtermékektől kiindulva az élek irányításával szembe járja be a maximális struktúrát. Olyan elemi reakciókat választ, melyektől vezet el végtermékig, vagyis amik előállítják a végtermékeket, kimerítő módon, az összes kombinációt tekintve. Egy-egy beválasztott elemi



2.5. ábra. Az RPISSG algoritmus egy kimenete a bután dehidrogénezése feladatra

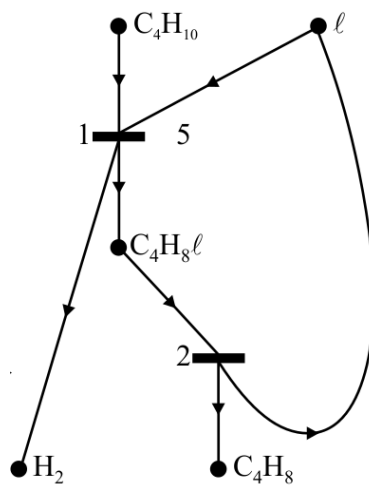
reakciólépésen keresztül újabb részecskéket ér el, melyeken továbbhaladhat a kiindulási anyagok felé. Egy részecskétől a hozzá vezető éleken haladhat tovább, majd rekurzívan hívja meg önmagát, hogy eldöntse az adott anyagot milyen kombináció állíthatja elő. Futása mindaddig tart, amíg megoldáshoz, vagy ellentmondáshoz nem jut.

2.6.3. RPIPBT

Az RPIPBT algoritmus egy reakcióút-szintézis feladatnak az axiómák által definiált összes megoldását generálja. Az összes megoldás megtalálásához egy keresési fa összes olyan ágát be kell járnia, ahol megoldás lehetséges, ezt visszalépéses elven folytatja. Működése hasonlít az RPISSG algoritmushoz. Kombinatorikusan lehetséges struktúrát generál, majd megvizsgálja, hogy fennáll-e az anyagmegmaradás, és a nulla eredőjű körmentesség. A lehetséges struktúrák generálását nem csak a végtermékek felől visszafele, hanem a kiindulási anyagoktól az élek irányítása mentén is végzi, így szimmetrikusan működik, mint az (R1), (R2), ..., (R5) axiómák. Futása addig tart, amíg van lehetséges megoldás, vagyis be nem járta az egész keresési fát.

Közvetlen utak generálása

Az RPIPBT algoritmusnak létezik olyan módosulata, amely csak a közvetlen utakat szolgáltatja. Közvetlen út alatt olyan lehetséges reakcióutakat értünk, melyekben egyetlen lépés sem áll elő a többi lépés semmilyen lineáris kombinációjával. Minden nem közvetlen út esetén az a lépés amely előáll több másik reakció lineáris kombinációjával helyettesíthető az adott lépéssel, így újra lehetséges reakcióutat kapunk. Minden nem közvetlen reakcióút tartalmaz egy közvetlen reakció utat. Ez a fajta algoritmusváltozat anyiban különbözik, hogy egy struktúra építése során korábban vizsgálja azokat a struktúrákat, melyeket részeként tartalmazhat.



2.6. ábra. Egy lehetséges direkt út, amit az RPIPBT algoritmus módosított változata generált

3. fejezet

Szétválasztási hálózat szintézis

3.1. Irodalmi áttekintés

A szétválasztási hálózatok rendkívül elterjedtek az iparban, és az élet számos területén, így fontosságukból kifolyólag a témához kötődő szakirodalom hatalmas. Több különböző feladat megfogalmazás, valamint megoldási módszer is ismert. A feladatok eltérhetnek betáplálási számban, termék minőségben, az alkalmazott matematikai modellben, a modell megoldási módjában és akár a szétválasztók modelljében is. A leggyakrabban alkalmazott matematikai modellek a lineáris (LP), nem-lineáris (NLP), egész változós lineáris (MILP) valamint az egész változós nem-lineáris (MINLP) modellek. Ezek megoldása pedig különböző (sztochasztikus, determinisztikus, közelítéssel) módszerekkel történik, attól függően mennyire pontos eredményt kívánunk elérni, és mik a korlátaink.

Termék minőség alatt azt értjük, hogy a termékek csak egy-egy komponenst tartalmaznak, vagy esetleg többet, tehát tiszta termékek, vagy kevert termékek. A szétválasztókat modelljük alapján több csoportba oszthatjuk. Az alapján, hogy milyen szétválasztást valósít meg beszélhetünk egyszerű, összetett valamint éles és nem éles szétválasztóról. Egy összetett szétválasztónak több bemenete, és több kimenete lehet, ezzel szemben egy egyszerűnek csak egy bemenete, és két kimenete. Egy adott komponens egy éles szétválasztó csak egy kimenetén jelenhet meg, amíg nem éles szétválasztó esetében több kimeneten is.

3.2. Alapfogalmak

Egy SNS feladat célja valamennyi, több komponensből álló, homogén folyam összetételének és nagyságának a megváltoztatása úgy, hogy azok teljesítsék a kívánalmakat. Folyam alatt olyan folytonosan haladó anyagot értünk, melynek nagyságát az idő függvényében is meg lehet adni (tömeg/idő, térfogat/idő mértékegységű), például folyamnak nevezhető egy csővezetékben mozgó valamilyen folyadék.

A feladat megoldása egy szétválasztási hálózat, melynek fő alkotórészei a szétválasztók, de tartalmazhat megosztókat, keverőket tárolókat és csővezetéseket is. A szétválasztók költségéhez képest a többi alkotórész költsége elhanyagolható, így egy hálózat költségét a benne szereplő szétválasztók összköltségével adjuk meg. Ábrázolás szempontjából érdemes ábrázolni a megosztókat és szétválasztókat is, mivel itt a folyamatok egyesülnek, illetve elágaznak. Az SNS feladatban felhasznált folyamatokat betáplálásnak, az előállítandó folyamatokat termékeknek nevezzük. Egy szétválasztóba bemenő folyamatot bemenetnek, a kilépő folyamatokat pedig kimeneteknek hívjuk. Jelen esetben csak egyszerű szétválasztókkal foglalkozunk, melyeknek egy bemenete és két kimenete van, ebben az esetben megkülönböztetünk alsó és felső kimenetet. Egy szétválasztás történhet a komponensek számos tulajdonsága alapján, például szemcseméret, illékonyság, másik anyagban történő old-

hatóság. A komponenseket sorba állíthatjuk a felhasznált tulajdonság alapján, így lehetőségünk van rájuk sorszámmal hivatkozni nevük helyett. Egy folyamat egy vektorral reprezentálunk úgy, hogy egy sorvektorban megadjuk a komponensek mennyiségét sorrendjük szerint. Egyszerű éles szétválasztók esetében azt mondjuk, hogy a szétválasztó az i -dik komponens után vág, ha bemenetét úgy képezi le, hogy az i -nél kisebb sorszámú komponensek az alsó, a nagyobbak pedig a felső kimenetbe kerülnek. Minden szétválasztónak van egy költségfüggvénye, mely megadja, az adott szétválasztón adott mennyiségű anyag szétválasztásának költségét.

A megosztó egy bemenettel, és kettő vagy több kimenettel rendelkezik. Az anyagokat oly módon bontja szét, hogy minden kimenetén a komponensek aránya megegyezik a bemenet arányával. Költségét nullának tekinthetjük, mivel felfogható, csak csővezetékek találkozásának, és azok költsége elhanyagolható a szétválasztó költsége mellett.

Egy keverő egy kimenettel, és kettő vagy több bemenettel rendelkezik. Kimenetén a bemenetek összege jelenik meg, a bemeneteket egyesíti. Költsége szintén nullának tekinthető, mivel legtöbb esetben csővezetékek egyesülése.

3.3. Szigorú szuperstruktúra, és redukált szigorú szuperstruktúra

Az SNS feladat megoldása két fő részből áll. Az elsőben egy struktúrális modellt generálunk, majd a második részben a modell alapján felírunk egy matematikai modellt, és annak a megoldása történik. A struktúrális modell alatt egy olyan modellt értünk, amely a felhasznált szétválasztókat, megosztókat, keverőket, és a köztük létesített kapcsolatokat tartalmazza. Jelentősége az, hogy a benne szereplő alkotóelemek szerepelhetnek az optimális megoldásban.

A struktúrális modellt gyakran szuperstruktúrának nevezik, mivel feltételezik, hogy tartalmazza az összes lehetséges struktúrát. Egy szuperstruktúrát akkor konstruálhatunk meg, ha ismerjük az összes lehetséges optimális struktúrát valamilyen módon. Az adott eset ritka, mivel egy betáplálás esetén is rohamosan növekszik a megoldások száma a komponensek számának növekedésével. Keresni kell tehát valamilyen közös tulajdonságot, amellyel minden lehetséges megoldásnak rendelkezni kell, de a tulajdonságra be kell látni, hogy valóban minden lehetséges struktúrának teljesítenie kell, és nem lehetséges struktúra nem teljesítheti. Ilyen tulajdonságot koránt sem egyszerű találni. Azonban bevezetésre került a szigorú szuperstruktúra fogalma, és ennek segítségével megkülönböztethető a helyesnek gondolt, és a bizonyítottan helyes szuperstruktúra. A szigorú szuperstruktúra annyival több elődjénél, hogy van definíciója, amely megköveteli, hogy tartalmazzon minden lehetséges optimális megoldást.

Definíció

Legyenek adottak a műveleti egységek halmazai a műveleti egységek modelljeivel együtt. Feltételezünk továbbá egy szisztematikus eljárást, mely az adott műveleti egységekből álló bármely hálózathoz olyan matematikai programozási modellt generál, amelynek megoldásával az optimális részhálózatát garantáltan meghatározza. Egy hálózatot szigorú szuperstruktúrának nevezünk, ha a feladatosztályába tartozó bármely feladatra ezen szigorú szuperstruktúra alapján generált matematikai modell megoldásánál kisebb költségű megoldás nem kapható más hálózat és modellgenerálás alapján.

Fontos megemlíteni, hogy adott feladathoz több szigorú szuperstruktúra is tartozhat, például ha egy meglevő szigorú szuperstruktúrát bővítünk új eszközökkel, és összeköttetésekkel, akkor szintén szigorú szuperstruktúrát kapunk. Ezáltal minden olyan struktúra amely tartalmazza a szigorú szuperstruktúrát, tartalmazni fogja az összes lehetségesen optimális struktúrát is. Ideális eset akkor állna elő, ha meg tudnánk

határozni a minimális szigorú szuperstruktúrát. Ha nem is a minimális szigorú szuperstruktúra generálásra, de szigorú szuperstruktúra generálásra bemutatjuk a következő algoritmust.

Az algoritmus menete a következő: bemeneteként megkapja a betáplálások, a termékek és a felhasználható szétválasztók halmazát. Első lépésként minden betápláláshoz felvesz egy megosztót, amit összeköt a hozzá tartozó bemenettel, és minden termék elé felvesz egy keverőt, és összeköti a megfelelő termék, keverő párokat. Az algoritmus második részét végrehajtja egyszer minden hálózatban levő, és megjelenő megosztóra. A végrehajtandó lépések az alábbiak: egy még nem vizsgált megosztó kiválasztása után megvizsgáljuk, hogy a rendelkezésre álló szétválasztók közül melyikek képesek lehetséges szétválasztást megvalósítani a kiválasztott megosztó kivezetésén.

Egy szétválasztás akkor lehetséges, ha a szétválasztó mindegyik kimenetén jelenik meg egy-egy folyam. Ha egy szétválasztó lehetséges szétválasztást valósít meg, akkor összekötjük a bemenetét a vizsgált megosztó kimenetével, majd a szétválasztó kimenetein új megosztókat hozunk létre, amiket majd később vizsgálunk ugyanezzel a módszerrel. Végezetül pedig kikerülő folyamatokat hozunk létre a vizsgált megosztóból minden olyan keverőbe, aminek a kimenete teljes egészében tartalmazza a vizsgált megosztó kimenetét. A kikerülő folyamatoknak az a jelentősége, hogy egyenesen belevezethetők egy keverőbe, így nem kell szétválasztani őket, teljes egészében felhasználhatók szétválasztás nélkül, így csökkentik az összköltséget. A művelet akkor ér véget, ha a bemenetekről származó megosztókat, és minden futás után keletkezett új megosztót is megvizsgáltunk egyszer végrehajtva rajta az algoritmus második lépését. A művelet sor eredménye a szigorú szuperstruktúra az adott feladathoz. További információ a szuperstruktúrával kapcsolatban, valamint annak a tételnek a bizonyítása, hogy minden feladatra ami egyszerű éles szétválasztókat tartalmaz létezik körmentes optimális megoldása abizonyítás megtalálható [12].

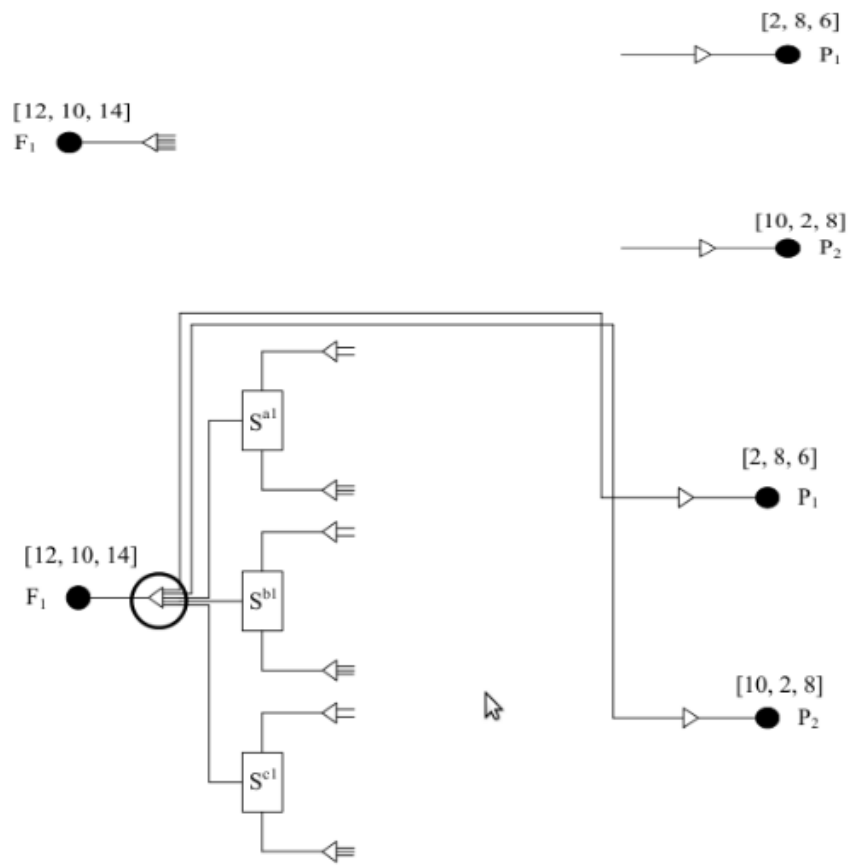
Ha megvizsgáljuk a kapott szigorú szuperstruktúrát, megállapítható, hogy nem minimális, mivel tartalmazhat azonos tulajdonsággal rendelkező szétválasztóból, vagy másik műveleti egységből többet is. Ideális eset akkor állna elő, ha meg tudnánk határozni a minimális szigorú szuperstruktúrát.

Az új struktúra alapja egy egyszerűsítő eljárás, amellyet a kapott szigorú szuperstruktúrán végezhetünk el. Az eljárás megkeresi és összevonja az azonos, összevonható szétválasztókat, így csökkentve az eredeti szigorú szuperstruktúra méretét. Összevonás két módszer alapján hajtható végre:

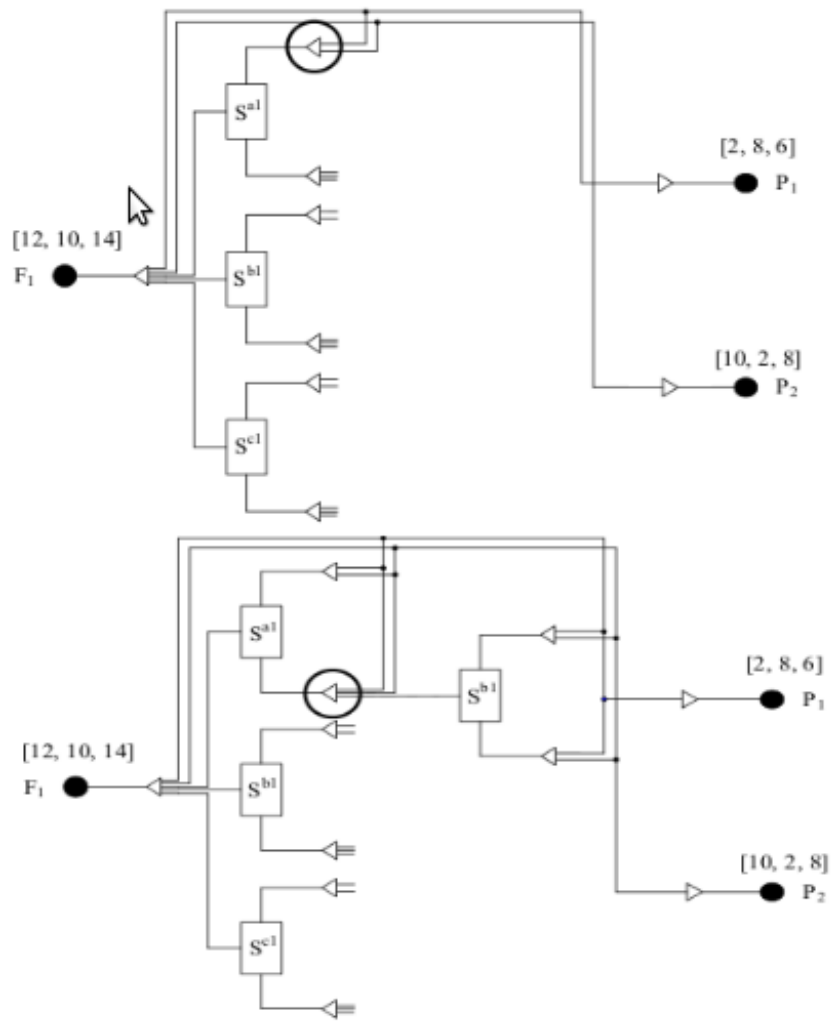
Az első módszer az azonos megosztási arányon alapul. Az összevonás azért lehetséges, mert lineáris arányos költségű szétválasztókkal dolgozunk, így két azonos típus összevonása egy nagyobb szétválasztóvá nem változtatja az összköltséget. A 3.3. ábra bal oldalán látható két szétválasztó ugyanúgy a 2. komponens után vág, így azok összevonhatóak, és helyettesíteni őket egy a jobb oldalon látható szétválasztóval lehet.

Második módszer az azonos bemeneti összetétel alapján történik. Ha találunk két azonos összetételű folyamatot a struktúrában, akkor ezeket egybe vezethetve szűkíthetjük a szigorú szuperstruktúrát. Az összevonás azért végezhető el, mert ha a két folyamat összetétele megegyezik, akkor abból mindkét helyen csak azonos felépítésű részstruktúra származtatható, ekkor a szigorú szuperstruktúránk tartalmazna azonos részeket. Az összköltség nem fog változni, mivel megint ugyanazokat az folyamatokat fogjuk feldolgozni, csak egy közös ágba, és a szétválasztás nem a szétválasztó méretétől, hanem a rajta szétválasztott mennyiségtől függ. A 3.3. ábrán bemutatott ötkomponenses folyamat szétválasztása közben azonos részgráfok jelennek meg, melyek összevonhatóak.

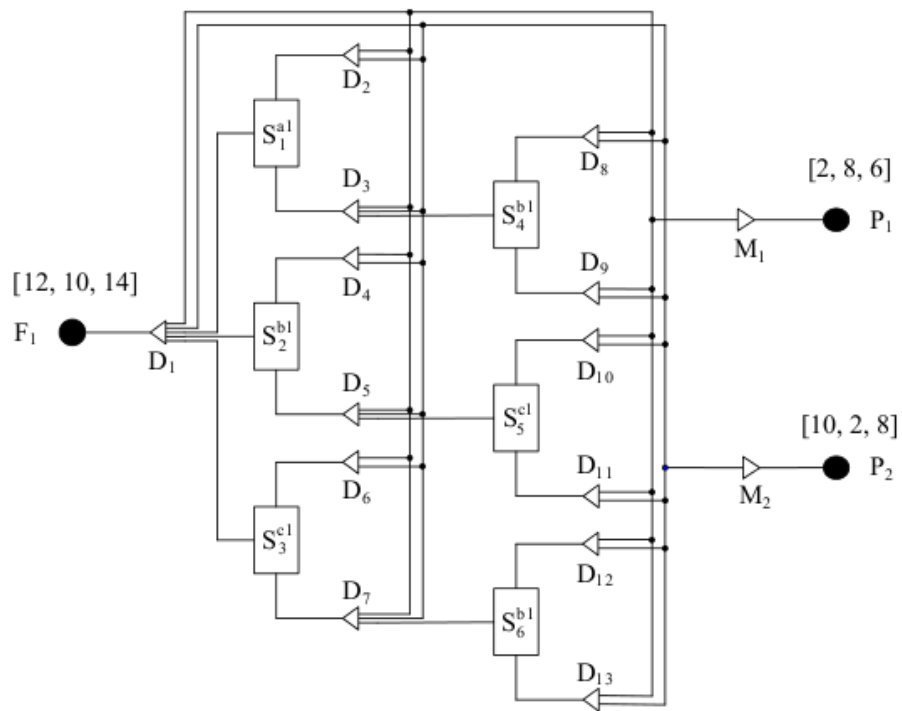
Redukált szuperstruktúrának nevezzük azt a szuperstruktúrát, amelyen már nem képes egyszerűsíteni a két eljárás egyike sem. Egy ilyen struktúrát úgy kaphatunk, hogy a feladatra először megadjuk a szigorú



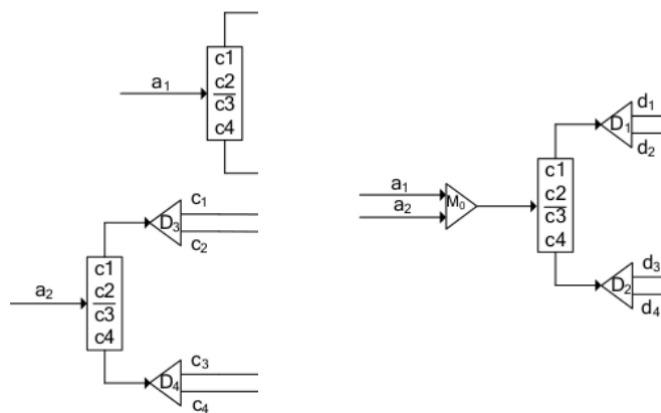
3.1. ábra. Szigorú szuperstruktúra generálás lépései



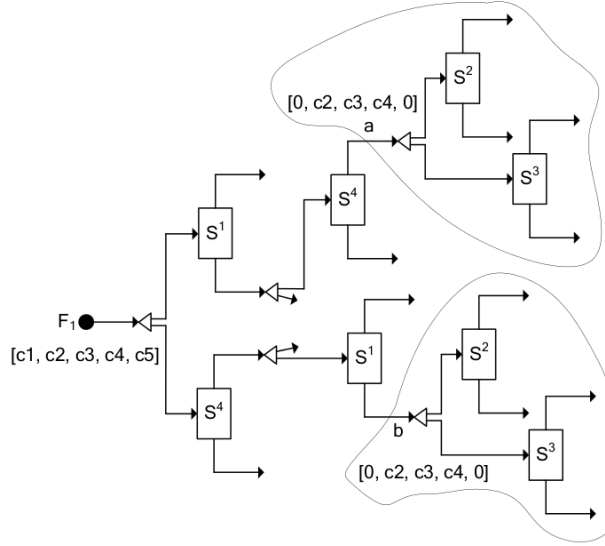
3.2. ábra. Szigorú szuperstruktúra generálás lépései



3.3. ábra. A generált szuperstruktúra



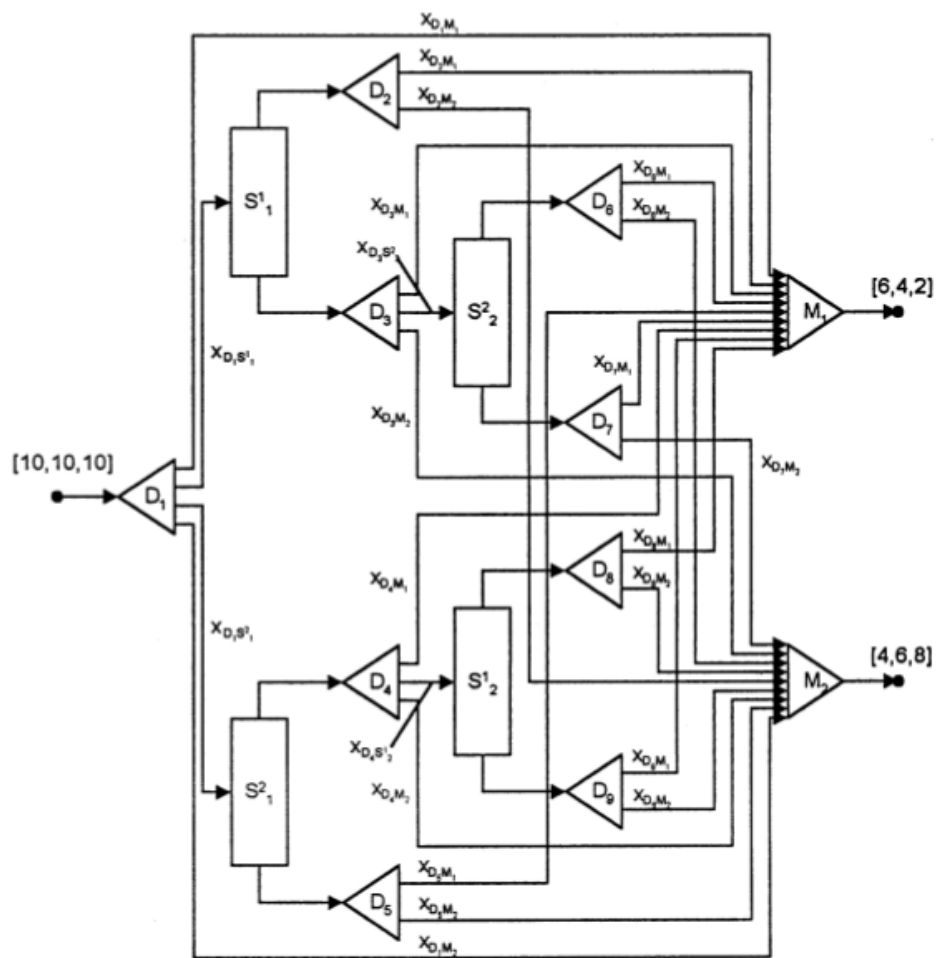
3.4. ábra. Azonos megosztási arány alapján történő összevonás



3.5. ábra. Azonos bemeneti összetétel alapján történő összevonás

szuperstruktúrát, majd a kapott struktúrát egyszerűsítjük ameddig csak lehet, valamint előálíthatjuk a bemeneti adatok ismeretében is hasonlóan mint a szigorú szuperstruktúrát.

A folyamat nagyon hasonló a szigorú szuperstruktúra generálásához. Kezdetben itt is felvesszünk megosztókat a bemenetekhez, és keverőket a termékekhez, majd ezek után ugyanúgy végig vizsgáljuk az összes eredeti és futás közben keletkező megosztót mind a szétválasztások, mind pedig az elkerülő folyamatok szerint. A különbség ott következik, ha egy szétválasztó képes lehetséges szétválasztást megvalósítani az adott megosztó kimenetén, akkor csak abban az esetben veszünk fel új megosztót a szétválasztó kimenetére, ha adott tulajdonságokkal rendelkező folyam még nem szerepel a hálózatban. Ha szerepel a hálózatban, akkor azt a kimenetét belevezetjük a már meglevő folyamba. Azonos tulajdonságú folyamatokon olyan folyamatokat értünk, melyek ugyanazokat a komponenseket tartalmazzák, ugyanolyan arányban (például A10B6 azonos a 2·A5B3-al).



3.6. ábra. Az 1.feladat redukált szigorú szuperstruktúrája

4. fejezet

SNS-RPI modelltranszformáció

A jelen fejezetben egy módszert adunk arra, hogy hogyan írhatunk fel egy olyan SNS feladatra RPI modellt, ami egyszerű, éles szétválasztókat tartalmaz egy szétválasztó osztályból. Ahoz, hogy az SNS feladatot RPI eszközökkel vizsgálhassuk, be kell látni hogy a két modell ekvivalens egymással. Meg kell vizsgálni, hogy fennállnak-e az RPI-nál tapasztalt megfigyelések, és strukturális tulajdonságok valamilyen formában az SNS feladatban is.

4.1. A két feladat ekvivalenciája

Vegyük észre, hogy a két feladat hasonlóságokat mutat, mint szerkezetben, mit pedig a struktúrák építőköveiben is. Egy-egy anyagra tekinthetünk úgy mint egy kémiai részecskére, valamint ha jobban végiggondoljuk az SNS feladat műveleti egységeire tekinthetünk ugyanúgy mint egy reakcióra, mivel mindkettő egy bementi halmazból állít elő egy kimeneti halmazt.

Továbbá vizsgáljuk az SNS szigorú superstruktúráját. A struktúrában bizonyítottan nem lehet recirkuláció [12], vagyis nem haladhat keresztül egy adott komponens kétszer egy adott műveleti egységen. A műveleti egységek definíciójából következik, hogy teljesül az anyagmegmaradás a struktúrában, ugyanis minden műveleti egységre igaz, hogy a bemenő folyamatok összege megegyezik a kimenő folyamatok összegével.

A fenti megállapítások, és alapvető elvárásokra építve felírhatunk egy aksiómahalmazt az RPI feladathoz hasonlóan, mely elemeinek minden esetben teljesülnie kell, ha egy általunk vizsgált szétválasztási feladatról van szó.

(RS1) Minden végtermék a kívánt mértékben kerül előállításra.

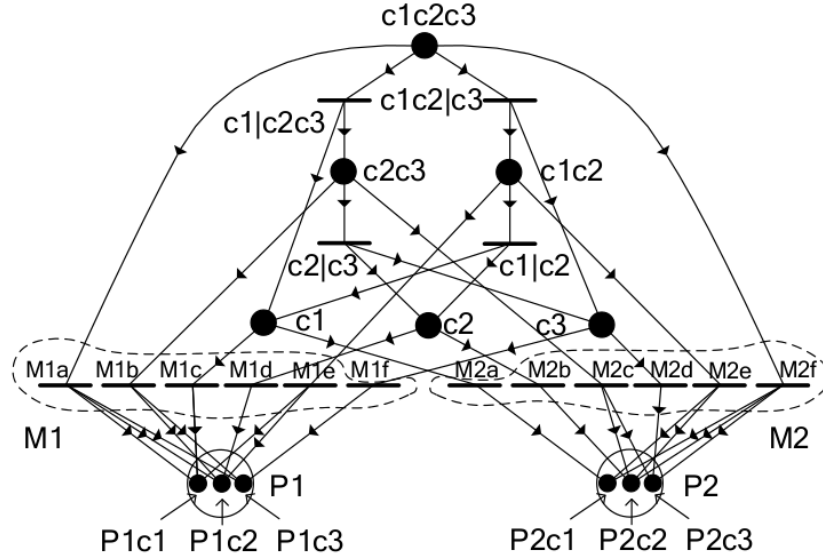
(RS2) Minden betáplálás felhasználásra kerül.

(RS3) A struktúrában szereplő valamely műveleti egység által előállított anyagot a struktúrában szereplő más műveleti egységek teljes mértékben felhasználják, illetve a struktúrában szereplő valamely műveleti egység által felhasznált minden anyagot a hálózatban szereplő más műveleti egységek teljes mértékben előállítják.

(RS4) A struktúrában szereplő minden szétválasztó előre meghatározott.

(RS5) A struktúra nem tartalmaz nulla eredőjű kört.

A fent említett aksiómahalmaz azonosítja azokat a hálózatokat melyeket lehetségesnek tekintünk.



4.1. ábra. Keverők és a termékek felsosztása több műveleti egységre, valamint komponensenkénti anyagokra

4.1.1. P-gráf reprezentáció az SNS feladathoz

Hasonló viselkedésük miatt tekinthetjük a szétválasztókat, mixereket és megosztókat műveleti egységeknek, ezért egy szuperstruktúrát ábrázolhatunk egy (m, o) P-gráffal, ahol m halmaz a hálózatban megjelenő anyagok halmaza, o pedig a műveleti egységek halmaza. Azonban ügyelnünk kell arra, hogy egy műveleti egység csak akkor működő képes, ha az összes bemeneti anyaga elő lett állítva korábban. Ahhoz, hogy ne okozzon problémát, a keverőket több műveleti egységgel valósítjuk meg, méghozzá oly módon, hogy a keverő minden bemenetéhez felvesszünk egy műveleti egységet, melynek egy bemente és egy kimenete van. Még egy megfontolás szükséges, mégpedig azért, mert csak a termékek mennyiségét ismerjük, magából az anyagnévből nem következtethetünk az összetételre. Ezt úgy küszöbölhetjük ki, ha egy terméket annyi anyaggal reprezentálunk, mint amennyi komponenszt tartalmaz, vagyis egy termék komponenseit külön anyagnak tekintjük. Példaként tekintsük a 4.1.1.ábrát.

A szigorú szuperstruktúra tulajdonságait vizsgálva, a P-gráf reprezentáció, és a megfogalmazott aximómákat követően megfogalmazhatunk két tételt, melyeknek teljesülnie kell minden lehetséges SNS feladatból képzett RPI modellben is.

Tétel

Minden lehetséges szigorú szuperstruktúrát reprezentáló (m, o) P-gráfban, bármely $a_j \in m$ anyagtól vezet út legalább egy végtermékig.

Bizonyítás:

Elsőnek vegyük a termékeket. Mivel minden termékből vezet 0 hosszúságú út önmagába, így rájuk igaz lesz.

Továbbá tegyük fel, hogy egy (m, o) lehetséges szuperstruktúrában létezik olyan $a_j \in m$ anyag, amelyből nem vezet út a gráfban egyetlen termékig sem. Jelölje p a gráfban az a_j anyagtól irányított úton elérhető részecskék halmazát, p' pedig a gráfban az a_j anyagtól irányított úton elérhető műveleti egységek halmazát.

Lemma:

p' tartalmazza az összes műveleti egységet, ami a p halmazban szereplő anyagokat felhasználja, illetve p tartalmazza az összes anyagot, amit a p' halmazban szereplő műveleti egységek előállítanak.

Lemma bizonyítása:

Az (RS2)-es és az (RS3)-as axióma szerint az a_j anyagot legalább egy műveleti egység felhasználja, így p' halmaz nem lehet üres. A p és p' halmazok definíciójából pedig az következik, hogy ha egy $a_k \in p$ anyagtól vezet el egy m_i műveleti egységig, akkor az a_j anyagtól vezet út az m_i műveleti egységig, tehát $m_i \in p'$, ha egy $m_i \in p'$ műveleti egységtől vezet el egy a_k anyagig, akkor az a_j anyagtól vezet út az a_k anyagig is, tehát $a_k \in p$.

Jelölje $f^+(p)$ és $f^-(p)$ a p halmazban szereplő anyagokból felhasznált és előállított anyagmennyiségeket. Az eredeti feltevés szerint p nem tartalmazza a végtermékeket, így ezen részecskékből az o halmazban szereplő műveleti egységek legalább annyit fel kell használniuk, mint amennyit előállítanak ($f^+(p) \geq f^-(p)$). Jelölje $g^-(p')$ és $g^+(p')$ a p' halmazban szereplő műveleti egységek által felhasznált és előállított anyagmennyiségeket. A lemma alapján a p halmaz elemeit legfeljebb a p' halmazban szereplő műveleti egységek használják fel, tehát $f^+(p) \leq g^-(p')$. Továbbá, a p' halmazban szereplő műveleti egységek a lemma szerint legfeljebb p elemeit állíthatják elő, tehát $g^+(p') \leq f^-(p')$.

$$g^-(p') \geq f^+(p) \geq f^-(p) \geq g^+(p')$$

Mivel a feladat megfogalmazásában feltettük az anyagmegmaradást a műveleti egységekben, így a műveleti egységek pontosan olyan mennyiségű anyagot használnak fel, mint állítanak elő.

$$g^-(p') = g^+(p')$$

Az előbbi egyenlőtlenségből, és az egyenlőségből következik, hogy:

$$g^-(p') = f^+(p) = f^-(p) = g^+(p').$$

Tehát a p' halmazban szereplő műveleti egységek a p halmazbeli anyagokat fogják azonos mértékben felhasználni és előállítani, vagyis nulla eredőjű kört alkotnak, így a vizsgált hálózat az (RS5) axióma szerint nem lehetséges, és ezzel ellentmondásra jutottunk.

Tétel

Minden lehetséges szigorú szuperstruktúrát reprezentáló (m, o) P-gráfban, bármely $a_j \in m$ anyagig vezet út legalább egy betáplálástól.

Bizonyítás:

A tétel bizonyítása hasonló módon történik mint az első tételé, csak a felhasznált és előállított, végtermék és betáplálás, anyagtól anyagig, végtermékig és betáplálástól kifejezéseket fel kell cserélni.

A két tétel bizonyítását követően beláttuk, hogy egy SNS feladatra felépített módosított szuperstruktúra elvárásaiban megegyezik egy RPI struktúrával, így az SNS feladathoz sikerült RPI ekvivalens struktúrát megadnunk.

4.2. Algoritmus

A következőkben bemutatom a modelltranszformációra kidolgozott algoritmust. A kitűzött cél egy olyan eljárás megvalósítása volt, amely egy SNS feladatra képes olyan különböző alternatív megoldásokat generálni, melyek költségben kissé térnek el a minimálistól, szerkezetileg viszont különbözőek. Gyakorlati szerepe akkor lehet, ha egy optimális szétválasztási hálózatot nem tudunk megvalósítani a gyártórendszer fizikai adottságai miatt, és ez a megoldás helyett szeretnénk keresni egy egy hozzá hasonló költségű, de eltérő topológiájú hálózatot, amely már megvalósítható. Ezt olyan formában valósítottam meg, hogy megvalósítottam és beláttam egy módszer helyességét amely az SNS feladatból RPI struktúrát generál, és a kapott RPI struktúrát ezek után megoldottam az RPI feladatokra konstruált algoritmussal (RPIPBT és módosítása). Jelen fejezetben bemutatom azt az algoritmust, amely előállítja az RPI struktúrát az SNS feladatból.

Az algoritmus felépítése a következő: Bemenerete az adott betáplálások, és termékek komponensenkénti anyagmennyisége, valamint a szétválasztók költség együtthatói. Itt fontos megjegyezni, hogy n komponens esetén $n - 1$ költséggyütthatót várunk, mivel n elem között $n - 1$ helyen történhet szétválasztás.

Két fő részből áll. Az első részben a bemeneti anyagokból képezzük azokat az anyagokat, amelyeket a redukált szuperstruktúra tartalmaz. Azért nem a redukált szuperstruktúrából indulunk ki, mivel egy műveleti egység működésképtelen akkor ha hiányzik egy bemenete, előre azonban nem tudhatjuk, hogy a generálás során mely műveleti egységekre nem lesz szükségünk, így kizárhatunk olyanokat amelyek egy másik műveleti egység bemenetét gyártaná le. Ezek az esetek kiköszöbölése bonyolultabb feladat lenne, mint a már meglévő anyagokból visszaírni, hogy melyik anyagot hogyan bonthatjuk fel. A feladatban csak éles szétválasztókat használunk, melyek úgy valósítják meg a szétválasztást, hogy egy adott komponensig az alsó, míg attól kezdve a felső kimeneteken jelentetik meg a komponenseket, és minden komponens csak a felső vagy az alsó kimeneten jelenik meg teljes egészében.

Abból kifolyólag, hogy csak éles szétválasztókat használunk, és a komponenseket egy adott tulajdonság alapján sorba rendezhetjük, így a megjelenő anyagok csak olyanok lehetnek, melyek egymás melletti komponenseket tartalmaznak.

Tétel

A hálózatban szereplő anyagok olyanok lehetnek, melyek csak egymás melletti komponenseket tartalmaznak, ettől eltérő eset csak akkor fordulhat elő, ha az adott bemeneten eredetileg sem szerepelt a hiányzó komponens.

Bizonyítás:

Tegyük fel, hogy keletkezik egy olyan anyag, amelyben nem egymás melletti komponensek szerepelnek. Két eset lehetséges. Az első az, ha a hiányzó komponensek az eredeti anyagból is hiányoztak. Ezt az esetet jónak tekinthetjük, mivel ha az eredeti anyagból is hiányoztak a komponensek, akkor a hiányzó komponensek létezésétől eltekintve a fennmaradó komponensek egymást követik. Ezzel az esettel a későbbiekben külön foglalkozunk, mivel egy speciális kiegészítő eset. A másik eset, ha a bemeneti anyagban szerepel, ám a jelenlegiben nem. Egy ilyen anyag előállítására egy szétválasztó képtelen a definíciója alapján, és a bemeneten nem jelenhet meg, mert azt az első esetben kezeljük. Csak úgy jöhet létre az anyag, hogy 2 vagy több anyagot egyesítünk, melyekből az adott komponens megelőző vágások során eltávolítottuk. Viszont ha az anyagot tovább bontjuk szétválasztókkal, akkor a kimeneteken meg fognak jelenni azok az anyagok, melyekből előállítottuk, így nulla eredőjű kört kapunk. Ellentmondásra jutottunk, mivel nulla eredőjű kört nem tartalmazhat

```

INPUT:  $(F, P, S, C)$ 
E :=  $(\{F\}, \{P\})$ ;
for all  $f \in F$  do
  for  $i = 1$  to  $|S|$  do
    for  $j = i$  to  $|S|$  do
       $d := \bigcup_{l=i}^j f_l$ ;
      if  $d \neq 0$  and  $k \cdot d \notin \mathcal{M}$  then
         $\mathcal{M} := \mathcal{M} \cup d$ ;
      end if
    end for
  end for
end for
 $\mathcal{M} := \mathcal{M} \cup F$ ;

for all  $m \in \mathcal{M}$  do
  for all  $s \in S$  do
     $(m, s) = d_1 + d_2$ ;
    if  $d_1 \neq 0$  and  $d_2 \neq 0$  then
       $O := O \cup (\{m\}, \{d_1, d_2\})$ ;
    end if
    for all  $p \in P$  do
      if BELEVIHETO  $(m, p)$  then
         $O := O \cup (\{m\}, \{p\})$ ;
      end if
    end for
  end for
end for
 $\mathcal{M} := \mathcal{M} \cup P$ ;
OUTPUT:  $(\mathbf{E}, O, \mathcal{M})$ 

```

4.2. ábra. A modelltranszformációt elvégző algoritmus pszeudo kódja

a hálózat.

Az összes ilyen tulajdonsággal rendelkező anyagot oly módon generálhatjuk, hogy mindig az eredeti anyag i -edik komponensétől kezdve képezünk új anyagot az ott található l egymást követő komponensekből ($1 \leq i \leq k$ és $1 \leq l \leq k$ ahol k a betáplálások komponensszáma). Ezt az anyagot viszont csak akkor tekinthetjük új anyagnak, ha nem szerepelt ezt megelőzően egy konstansszorosa sem. Például egy A5B4C2 anyag megegyezik egy A10B8C2 anyaggal, mivel felítható úgy mint $2 \cdot \text{A5B4C}$. Abból a megfontolásból nem tartjuk meg az összes anyagot, mert így jelentősen egyszerűsíthetjük struktúránkat, ami pedig a feladat megoldás idejét csökkenti később.

A második részben az előállított összes anyagról eldöntjük, hogy melyik szétválasztóval valósítható meg az anyagon szétválasztás, és hogy melyik termékbe vezethető bele közvetlenül az adott anyag, majd minden lehetséges esetre képezünk egy RPI elemi reakciónak megfelelő műveleti egységet a bemenő elemből és a kimenő elemekből. Egy szétválasztás akkor valósítható meg egy anyag k . komponense után, ha a szétválasztó alsó és felső kimenetén is jelenik meg anyag. Egy anyag pedig akkor vihető bele közvetlenül egy termékbe, ha a termék tartalmazza legalább az adott anyag minden komponensét, ennek a vizsgálatát végzi a BELEVIHETO() eljárás. A végső lépésként a transzformáció elvégzése után az SNS feladatból képzett RPI feladat kiírásra kerül.

```

BELEVIHETŐ  $(m, p)$ 
for all  $c \in m$  do
  if  $k \cdot c \notin p$  then
    return false;
  end if
end for
return true;

```

4.3. ábra. A BELEVIHETO eljárás pszeudo kódja

F	betáplálások véges halmaza
P	termékek véges halmaza
S	szétválasztók véges halmaza
C	komponensek halmaza
E	eredő reakció
O	elemi reakciók véges halmaza
$\mathcal{M}$	anyagok halmaza
f_l	az f anyag l -edik komponense
k	valós szám
i, j, l	egész szám
c	egy komponens $\in C$
d, p, d_1, d_2	anyagok $\in \mathcal{M}$
$(m, s) = d_1 + d_2$	Egy s szétválasztó az m anyagot d_1 és d_2 anyagokra bontja.
$(\{A\}, \{B\})$	Egy reakciót reprezentál, ahol az A halmaz elemeiből B halmaz elemei keletkeznek.
BELEVIHETO (m, p)	Egy függvény, mely logikai értékkel tér vissza. Igazzal, ha m anyag belevihető közvetlenül p termékben, ami azt jelenti, hogy m minden egyes komponense szerepel p ben is. Hamissal, ha nem vihető bele.

4.4. ábra. Jelölések a pszeudo kódban

4.3. További tulajdonságok

Az modelltranszformáció véges időben véget ér. Az algoritmus második felében csak az első részben legenerált anyagok számától függ az elvégzendő műveletek száma, mivel a szétválasztók, bemenetek és termékek számát előre tudjuk, így azokat tekinthetjük konstansnak. Elég azt belátni, hogy véges sok anyagot generálunk csak. Ezt annyiból is megtehetjük hogy egy k komponensű anyagra biztosan kevesebb anyagot generálunk mint $(k+1)^k$ ami a k darab komponens és a 0 (amikor nem szerepel egy komponenes) ismétléses variációinak száma. A továbbiakban azt vizsgáljuk adható-e kisebb korlát a generált anyagok számára. Azt a tulajdonságot kívánjuk kihasználni, hogy generált anyag csak olyan lehet ami egymás melletti komponenseket tartalmaz csak (ha egy komponens eredetileg 0 volt, most ettől eltekinthetünk, mert meg volt adva, és az a komponenes is benne van az anyagban, csak 0 együtthatóval). Egy 0 darab komponenes nem követheti egymást, mert akkor valójában nem is beszélünk anyagról. A legrövidebb anyag 1 komponensből áll, igaz nem követi egymást több komponens, de tekinthetjük jónak, mivel úgy kapjuk, hogy egy legalább 2 hosszú anyagot elválasztunk. A leghosszabb egymást követő komponensek száma maximum k lehet mivel összesen ennyi különböző komponenesünk van. Olyan anyag ami l komponenset tartalmaz a k közül összesen annyi lehet, mint amennyi a fennmaradó komponensek száma $(k-l)$ plussz az adott anyag. Mivel az l hosszú résznek mindig egymást kell követnie, így maximum $k-l$ helyre tolható el. Tehát olyan anyag, ami l komponenset tartalmaz k közül $k-l+1$ darab van. Az összes anyag számát a következő képp írhatjuk fel ezek után:

$$\sum_{i=1}^k k-i+1 \quad (4.1)$$

$$\sum_{i=1}^k k-i+1 = \sum_{i=1}^k k - \sum_{i=1}^k i + \sum_{i=1}^k 1 \quad (4.2)$$

Minden résztag összegét ismerjük, az első nem más mint k^2 , a második egymást követő számok össze 1-től k -ig, ami $\frac{k(k+1)}{2}$, a harmadik pedig $k \cdot 1$. Ezeket felhasználva megadhatjuk a teljes összeget is.

$$\sum_{i=1}^k k - \sum_{i=1}^k i + \sum_{i=1}^k 1 = k^2 - \frac{k(k+1)}{2} + k = \frac{k(k+1)}{2} \quad (4.3)$$

Tehát az algoritmus első része által generált anyagok száma egy k komponensű anyag esetén:

$$\sum_{i=1}^k k-i+1 = \frac{k(k+1)}{2} \quad (4.4)$$

Több bemenet esetén a bemenetek összetételétől függ, de legrosszabb esetben is $\frac{m \cdot k(k+1)}{2}$ ahol m a bemenetek számát jelöli. A legrosszabb eset akkor áll fent, ha minden anyagban minden komponens mennyiség együtthatója különböző príszám. Ebben az esetben csak az 1 hosszú anyagok egyszerűsíthetők, a többi nem, így a felső korlát valójában $\frac{m \cdot k(k+1)}{2} - (m-1)k$

A generált anyagok számának meghatározása után meghatározzuk a szükséges szétválasztók számát is. Egy l komponenset tartalmazó anyagot maximum $l-1$ helyen lehet szétválasztani, és egy l komponenesből

álló anyagból $k - l + 1$ darab van, így az szükséges szétválasztók száma egy bemenet esetén:

$$\sum_{i=1}^k (i-1)(k-i+1) \quad (4.5)$$

Felbontva a zárójeleket, és szétbontva több részre:

$$\sum_{i=1}^k (i-1)(k-i+1) = \sum_{i=1}^k i \cdot k + 2 \cdot i - i^2 + k - 1 \quad (4.6)$$

$$\sum_{i=1}^k i \cdot k + 2 \cdot i - i^2 + k - 1 = k \cdot \sum_{i=1}^k i + 2 \cdot \sum_{i=1}^k i - \sum_{i=1}^k i^2 + \sum_{i=1}^k k - \sum_{i=1}^k 1 \quad (4.7)$$

Minden külön szumma értékét ismerjük pontosan, párat az előbb már említettünk, a fentmaradó 3. pedig nem más mint a négyzetszámok összege 1-től k -ig, ami $\frac{k(k+1)(2k+1)}{6}$. A szummákat kibontva, majd összegezve magkapjuk a végeredményt.

$$\sum_{i=1}^k (i-1)(k-i+1) = \frac{k^2(k+1)}{2} + \frac{2k(k+1)}{2} - \frac{k(k+1)(2k+1)}{6} + k^2 - k \quad (4.8)$$

Közös nevezőre hozás, a zárójelek kibontása és a lehetséges egyszerűsítések elvégzését követően:

$$\sum_{i=1}^k (i-1)(k-i+1) = \frac{k^3 - k}{6} \quad (4.9)$$

Tehát egy k komponensű bemenet esetén a szükséges szétválasztások száma $\frac{k^3-k}{6}$. Több bemenet esetén a számuk szintén az anyagok összetételétől függ, de legrosszabb esetben is $\frac{m \cdot (k^3-k)}{6}$ ahol m a bementek számát jelöli. A legrosszabb eset itt is a már említett esetben lesz.

4.4. Megvalósítás

A modelltranszformációra kidolgozott algoritmust megvalósítottam C++ programozási nyelven. Ebben a fejezetben az implementálás során felmerült, és általam megoldott érdekes részproblémákat mutatom be.

Az első ilyen probléma volt, az algoritmus első felében annak felismerése, hogy egy adott anyag konstans-szorosa szerepelt-e már előzőleg. Két folyam akkor is megegyezhet, ha nem minden komponensből ugyanannyit tartalmaznak, de minden komponens azonos arányban tartalmaznak. Ezt úgy oldottam meg, hogy minden anyaghoz hozzárendeltem a komponensek mennyiségeinek legnagyobb közös osztóját, majd a legnagyobb közös osztóval leosztva minden komponens az adott anyagból olyan anyagot generáltam, amely nem egyszerűsíthető tovább. Ezáltal a generált anyagok szimplán összehasonlíthatók, mert egyik sem egyszerűsíthető tovább. A legnagyobb közös osztó meghatározásához az Euklidész által megadott legnagyobb közös osztó meghatározására szolgáló algoritmust oly módon módosítottam, hogy az alkalmas legyen n szám legnagyobb közös osztójának meghatározására.

A másik érdekes probléma az algoritmus második felében annak eldöntése, hogy egy adott anyagon melyik szétválasztó tud szétválasztást megvalósítani. A szétválasztás ott okoz problémát, hogy egy szétválasztó csak akkor tud létesíteni szétválasztást, ha mindegyik kimenetén jelenik meg anyag, azonban különböző komponensszámot tartalmazó anyagokkal, és több szétválasztóval a probléma mérete nő. Ezt úgy oldottam meg,

hogy mindig előre generáltam azokat az anyagokat amiket a szétválasztó kimenetein eredményezne, és ha volt köztük olyan, amely nem tartalmaz komponenst, akkor nem valósítottam meg az adott szétválasztást az adott anyagra.

Tovább lépési lehetőségek

A közeljövőben tervezem kiterjeszteni a konverziós módszert olyan SNS feladatokra, amelyek tartalmazhatnak több szétválasztó családot is. Az olyan feladatokra is igazak lesznek a megállapítások, mivel csak annyiban különböznek, hogy többfajta szétválasztót tartalmaz, így a generált RPI feladat újból ekvivalens lesz az eredeti SNS-el. Azonban az algoritmus több új változtatást is igényel, mivel két külön osztályba tartozó szétválasztó más-más tulajdonság alapján hajtja végre a szétválasztást, így más lesz a komponenssorrend minden egyes külön családból származó szétválasztó esetén.

5. fejezet

Megoldott feladatok

A bemutatott modelltranszformációs eljárást számos szakirodalomból ismert példán kipróbáltuk, majd a kapott új modellt megoldottam. Az módszer hatékonyságát mutatja, hogy a kapott eredmények megegyeztek, vagy kisebbek voltak az irodalomból ismert eredményeknél, és ezek alapján számos optimálisához közeli megoldást is kaptunk. A következő részfejezetben a megoldott feladatok közül mutatok be hármat, mind a három feladat korábban publikálásra került [15]. A feladatok megoldásához Intel Pentium M 1.8GHz processzorral és 2 gigabájt memóriával felszerelt számítógépet használtam.

5.1. 1. Feladat

Az első feladat három komponensből álló folyamatból áll. Két többkomponensű terméket szeretnénk megkapni egy betáplálásból. Az 1. feladat adatait a 5.1.ábrán láthatjuk. Az első sorban levő A,B,... nagybetűk reprezentálják az 1.,2.,... komponenseket, a betáplálások, és a termékek soraiban az adott komponensből tartalmazott mennyiség található, a legalsó sorban a költség pedig azt adja meg, hogy mekkora költséggel vágható az adott anyag a töle balra, és jobbra levő komponensek között (ezek a jelölések a továbbiakban a 2. és a 3. feladatokra is igazak lesznek).

Komponensek	A	B	C
Betáplálás	10	10	10
Termék 1	6	4	2
Termék 2	4	6	8
Költség	1	1	

5.1. ábra. Az 1.feladat adatai

A modelltranszformációt megvalósító programot lefuttattam, majd a generált RPI modellt megoldottam RPI feladatnál bemutatott algoritmust megvalósító megoldóval. A kapott eredményeket megtekinthetjük a 5.2.ábrán összehasonlítva a [15]-ben publikált eredményekkel. Jelen esetben az optimális megoldás 12,ami megegyezik a [15] ben publikáltal. A következő struktúráisan nagymértékben eltérő, optimálisához legközelebb eső megoldás 16 költségű, ebből azonban 4 darab különbözőt is generált. Összesen 63 darab különböző megoldást generált hozzá, a többbit is érdemes megemlíteni, de ezek költsége egyre jobban eltér az optimálistól. A 63 darab különböző megoldást 0.386 másodperc alatt szolgáltatatta a megoldó program.

	Megoldási idő	Optimális költség
[15]-ben publikált	0.13 sec	12
Jelen megoldás	0.021 sec	12

5.2. ábra. Az 1.feladat eredményeinek összehasonlítása

5.2. 2. Feladat

A második feladat négy komponenst tartalmazó folyamatokból áll. Két többkomponensű terméket szeretnénk megkapni egy betáplálásból. A 2. feladat adatait a 5.3.ábrán láthatjuk.

Komponensek	A	B	C	D
Betáplálás	15	20	10	15
Termék 1	5	10	4	10
Termék 2	10	10	6	5
Költség	2.5	3.0	1.5	

5.3. ábra. A 2.feladat adatai

A feladat adataiból generált új RPI modell megoldásából származó eredményeket 5.4.ábrán láthatjuk a [15] eredményeivel összevetve. Egy jobb megoldást kaptam mint az általuk publikált 55.5. Ennek a feladatnak lényegesen több megoldása létezik, mint az előzőnek, így az optimálistól kismértékben eltérő megoldásokra bevezettem egy elvi, költségileg merően eltérő, de nem drasztikus határt, ami alatt megemlítem őket. Ezt a határt 20%-ban határozzuk meg. Tehát csak olyan megoldásokat tekintünk, melyek maximum 20%-al térnek el az optimális megoldástól. Ilyen megoldást 28 darabot generált a megoldó 0.382 másodperc alatt.

	Megoldási idő	Optimális költség
[15]-ben publikált	1.37 sec	55.50
Jelen megoldás	0.036 sec	54.25

5.4. ábra. Az 2.feladat eredményeinek összehasonlítása

5.3. 3. Feladat

A harmadik feladat négy komponenst tartalmazó folyamatokból áll. Három többkomponensű terméket szeretnénk megkapni három betáplálásból. A 3. feladat adatait a 5.5.ábrán láthatjuk.

Jelen esetben is legeneráltam a feladat adatai alapján az RPI modellt, ami 21 anyagot, és 35 műveleti egységet tartalmazott, majd megoldottam a korábban említett módon. Az eredményeket összehasonlító táblázatot az 5.5.ábrán láthatjuk.

Azonban több mint 24%-al jobb megoldást szolgáltatott a megoldó a publikált eredménynél, így a 20%-os korláttól eltekintve, az összes olyan eredményt megadtam, amelyek összköltsége kisebb a publikált 138.18-nál. Összesen 1140 darab jobb megoldást volt, melyek generálása 1 perc 6.77 másodpercet vett igénybe (átlagosan 0.0587 másodperc/megoldás).

Komponensek	A	B	C	D
Betáplálás 1	6	4	0	0
Betáplálás 2	8	6	10	6
Betáplálás 3	0	0	5	5
Termék 1	10	3	2	0
Termék 2	4	7	7	2
Termék 3	0	0	6	9
Költség	4.0	1.5	4.0	

5.5. ábra. A 3.feladat adatai

	Megoldási idő	Optimális költség
[15]-ben publikált	0.74 sec	138.18
Jelen megoldás	0.041 sec	104.26

5.6. ábra. Az 3.feladat eredményeinek összehasonlítása

6. fejezet

Összefoglalás

Munkám során kifejlesztettem egy módszert, mely képes alternatív optimális megoldások generálására egy olyan szétválasztási hálózat szintézis problémára, amely egyszerű, éles szétválasztókat tartalmaz. Ezt a feladatot az SNS probléma RPI feladatosztályba való modelltranszformációjával értem el. Saját algoritmust dolgoztam ki a konverzió elvégzésére, valamint beláttam helyességét, és a generált struktúra ekvivalenciáját. Az algoritmust implementáltam C++ programozási nyelven. A módszert teszteltem számos szakirodalomban megtalálható példára, és minden esetben legalább a publikált optimális, vagy annál jobb megoldást kaptam számos optimálishoz közeli, de attól eltérő topológiájú megoldás mellett. Jövőbeli terveim közt szerepel a konverzió kiterjesztése olyan SNS feladatokra is, melyek több szétválasztó családból származó szétválasztót is tartalmazhatnak.

Irodalomjegyzék

- [1] Seo, H., D.-Y. Lee, S. Park, L. T. Fan, S. Shafie, B. Bertok, and F. Friedler, Graph-Theoretical Identification of Pathways for Biochemical Reactions, *Biotechnology Letters*, 23, 1551-1557 (2001)
- [2] Fan, L. T., B. Bertok, F. Friedler, and S. Shafie, Mechanisms of Ammonia-Synthesis Reaction Revisited with the Aid of a Novel Graph-Theoretic Method for Determining Candidate Mechanisms in Deriving the Rate Law of a Catalytic Reaction, *Hungarian Journal of Industrial Chemistry*, 29, 71-80 (2001)
- [3] Fan, L. T., B. Bertok, and F. Friedler, A Graph-Theoretic Method to Identify Candidate Mechanisms for Deriving the Rate Law of a Catalytic Reaction, *Computers and Chemistry*, 26, 265-292 (2002)
- [4] Fan, L. T., S. Shafie, B. Bertok, F. Friedler, D.-Y. Lee, H. Seo, S. Park, and S.-Y. Lee, Graph-Theoretic Approach for Identifying Catalytic or Metabolic Pathways, *Journal of the Chinese Institute of Engineers*, 28, 1021-1037 (2005)
- [5] Fan, L.T., Y.-C. Lin, S. Shafie, K. L. Hohn, B. Bertok, and F. Friedler, Comment on: An improved microkinetic model for the water-gas shift reaction on copper *Surf. Sci.* 541 (2003) 21-30, *Surf. Sci.*, 601, 2401-2405 (2007)
- [6] Lin, Y.-C., L. T. Fan, S. Shafie, K. L. Hohn, B. Bertok, and F. Friedler, Catalytic Pathways Identification for Partial Oxidation of Methanol on Copper-Zinc Catalysts: $\text{CH}_3\text{OH} + 1/2 \text{O}_2 \rightleftharpoons \text{CO}_2 + 2 \text{H}_2$, *Ind. Eng. Chem. Res.*, 47, 2523-2527 (2008)
- [7] Lin, Y.-C., L. T. Fan, S. Shafie, B. Bertok, and F. Friedler, Graph-theoretic Approach to the Catalytic-Pathway Identification of Methanol Decomposition, *Computer Aided Chemical Engineering*, 26, 1293-1298 (2009)
- [8] Lin, Y.-C., L. T. Fan, S. Shafie, B. Bertok, and F. Friedler, Generation of Light Hydrocarbons through Fischer-Tropsch Synthesis: Identification of Potentially Dominant Catalytic Pathways via the Graph-Theoretic Method and Energetic Analysis, *Comput. Chem. Eng.*, 33, 1182-1186 (2009)
- [9] Lee, D.-Y., L. T. Fan, S. Park, S. Y. Lee, S. Shafie, B. Bertok, and F. Friedler, Complementary Identification of Multiple Flux Distributions and Multiple Metabolic Pathways, *Metab. Eng.*, 7, 182-200 (2005)
- [10] Fishtik, I., A. Alexander, R. Datta, Enumeration and discrimination of mechanisms in heterogeneous catalysis based on response reactions and unity bond index-quadratic exponential potential (UBI-QEP) method. *Surface Science* 430,1-17(1999)
- [11] Happel, J., P.H. Sellers Analysis of the Possible Mechanisms for a Catalytic Reaction System *Adv.Catal.* 32,273-323(1983)

- [12] Kovács, Z. , Z. Ercsey , F. Friedler , L.T. Fan, Separation-network synthesis: global optimum through rigorous super-structure Computers and Chemical Engineering 24, 1881-1900 (2000)
- [13] Heckl,I. , F.Friedler , L.T. Fan , Reduced super-structure for a separation network comprising separators effected by different methods of separation , Computers and Chemical Engineering 33,687-698 (2009)
- [14] Fan, L.T. , T. Zhang , J. Liu , J. R. Schlup , P.A. Seib , F. Friedler , B. Bertok , Assessment of Sustainability-Potential: Hierarchial Approach Ind.Eng.Chem.Res. 46,4506-4516 (2007)
- [15] Quesada,I. , I.E.Grossmann , Global optimization of bilinear process networks with multicomponent flows , Comput. Chem. Eng. 19, 1219-1242 (1995)

7. fejezet

Függelék

7.1. A programhoz tartozó bemeneti, és generált modell

7.1.1. Az 1.feladat bemenete

```
Number of Components :  
3  
  
Number of Feed-streams :  
1  
  
Number of Product-streams :  
2  
  
Feed-streams :  
(10,10,10)  
  
Product-streams :  
(6,4,2)  
(4,6,8)  
  
Degrees of Difficulties :  
(1,1)
```

`sources/be1.sns`

7.1.2. Az 1.feladat kimenete

```
species :  
A10  
A10B10  
A10B10C10  
B10
```

B10C10

C10

P1A

P1B

P1C

P2A

P2B

P2C

overall reaction:

R1:A10B10C10=6 P1A + 4 P1B + 2 P1C + 4 P2A + 6 P2B + 8 P2C

elementary reactions:

r1 [20,0]:A10B10=>A10 + B10

r2 [30,0]:A10B10C10=>A10 + B10C10

r3 [30,0]:A10B10C10=>A10B10 + C10

r4 [20,0]:B10C10=>B10 + C10

r5 [0,0]:A10=>10 P1A

r6 [0,0]:A10=>10 P2A

r7 [0,0]:A10B10=>10 P1A + 10 P1B

r8 [0,0]:A10B10=>10 P2A + 10 P2B

r9 [0,0]:A10B10C10=>10 P1A + 10 P1B + 10 P1C

r10 [0,0]:A10B10C10=>10 P2A + 10 P2B + 10 P2C

r11 [0,0]:B10=>10 P1B

r12 [0,0]:B10=>10 P2B

r13 [0,0]:B10C10=>10 P1B + 10 P1C

r14 [0,0]:B10C10=>10 P2B + 10 P2C

r15 [0,0]:C10=>10 P1C

r16 [0,0]:C10=>10 P2C

sources/ki1.in

7.1.3. Az 2.feladat bemenete

Number of Components:

4

Number of Feed-streams:

1

Number of Product-streams:

2

Feed-streams:

(15,20,10,15)

Product-streams:

(5,10,4,10)

(10,10,6,5)

Degrees of Difficulties:

(2.5,3,1.5)

sources/be2.sns

7.1.4. Az 2.feladat kimenete

species:

A15

A15B20

A15B20C10

A15B20C10D15

B20

B20C10

B20C10D15

C10

C10D15

D15

P1A

P1B

P1C

P1D

P2A

P2B

P2C

P2D

overall reaction:

R1: A15B20C10D15=5 P1A + 10 P1B + 4 P1C + 10 P1D + 10 P2A + 10 P2B + 6 P2C + 5 P2D

elementary reactions:

r1 [87.5,0]: A15B20 $\Rightarrow$ A15 + B20

r2 [112.5,0]: A15B20C10 $\Rightarrow$ A15 + B20C10

r3 [135,0]: A15B20C10 $\Rightarrow$ A15B20 + C10

r4 [150,0]: A15B20C10D15 $\Rightarrow$ A15 + B20C10D15

r5 [180,0]: A15B20C10D15 $\Rightarrow$ A15B20 + C10D15

r6 [90,0]: A15B20C10D15 $\Rightarrow$ A15B20C10 + D15

r7 [90,0]: B20C10 $\Rightarrow$ B20 + C10

r8 [135,0]: B20C10D15 $\Rightarrow$ B20 + C10D15

```

r9 [67.5,0]: B20C10D15=>B20C10 + D15
r10 [37.5,0]: C10D15=>C10 + D15
r11 [0,0]: A15=>15 P1A
r12 [0,0]: A15=>15 P2A
r13 [0,0]: A15B20=>15 P1A + 20 P1B
r14 [0,0]: A15B20=>15 P2A + 20 P2B
r15 [0,0]: A15B20C10=>15 P1A + 20 P1B + 10 P1C
r16 [0,0]: A15B20C10=>15 P2A + 20 P2B + 10 P2C
r17 [0,0]: A15B20C10D15=>15 P1A + 20 P1B + 10 P1C + 15 P1D
r18 [0,0]: A15B20C10D15=>15 P2A + 20 P2B + 10 P2C + 15 P2D
r19 [0,0]: B20=>20 P1B
r20 [0,0]: B20=>20 P2B
r21 [0,0]: B20C10=>20 P1B + 10 P1C
r22 [0,0]: B20C10=>20 P2B + 10 P2C
r23 [0,0]: B20C10D15=>20 P1B + 10 P1C + 15 P1D
r24 [0,0]: B20C10D15=>20 P2B + 10 P2C + 15 P2D
r25 [0,0]: C10=>10 P1C
r26 [0,0]: C10=>10 P2C
r27 [0,0]: C10D15=>10 P1C + 15 P1D
r28 [0,0]: C10D15=>10 P2C + 15 P2D
r29 [0,0]: D15=>15 P1D
r30 [0,0]: D15=>15 P2D

```

sources/ki2.in

7.1.5. Az 3.feladat bemenete

Number of Components:

4

Number of Feed-streams:

3

Number of Product-streams:

3

Feed-streams:

(6,4,0,0)

(8,6,10,6)

(0,0,5,5)

Product-streams:

(10,3,2,0)

(4,7,7,2)

(0,0,6,9)

Degrees of Difficulties :
(4,1.5,4)

sources/be3.sns

7.1.6. Az 3.feladat kimenete

species :

A6

A6B4

B4

A8B6

A8B6C10

A8B6C10D6

B6C10

B6C10D6

C10

C10D6

D6

C5D5

P1A

P1B

P1C

P2A

P2B

P2C

P2D

P3C

P3D

overall reaction :

R1: A6B4 + A8B6C10D6 + C5D5 = 10 P1A + 3 P1B + 2 P1C + 4 P2A + 7 P2B + 7 P2C + 2 P2D + 6 P3C + 9 P3D

elementary reactions :

r1 [40,0]: A6B4 $\Rightarrow$ A6 + B4

r2 [336,0]: 6 A8B6 $\Rightarrow$ 8 A6 + 9 B4

r3 [288,0]: 3 A8B6C10 $\Rightarrow$ 4 A6 + 3 B6C10

r4 [36,0]: A8B6C10 $\Rightarrow$ A8B6 + C10

r5 [360,0]: 3 A8B6C10D6 $\Rightarrow$ 4 A6 + 3 B6C10D6

r6 [45,0]: A8B6C10D6 $\Rightarrow$ A8B6 + C10D6

r7 [120,0]: A8B6C10D6 $\Rightarrow$ A8B6C10 + D6

r8 [48,0]: 2 B6C10 $\Rightarrow$ 3 B4 + 2 C10

r9 [66,0]: 2 B6C10D6 $\Rightarrow$ 3 B4 + 2 C10D6

```

r10 [8 8 , 0]: B6C10D6=>B6C10 + D6
r11 [6 4 , 0]: C10D6=>C10 + D6
r12 [240 , 0]: 6 C5D5=>3 C10 + 5 D6
r13 [0 , 0]: A6=>6 P1A
r14 [0 , 0]: A6=>6 P2A
r15 [0 , 0]: A6B4=>6 P1A + 4 P1B
r16 [0 , 0]: A6B4=>6 P2A + 4 P2B
r17 [0 , 0]: B4=>4 P1B
r18 [0 , 0]: B4=>4 P2B
r19 [0 , 0]: A8B6=>8 P1A + 6 P1B
r20 [0 , 0]: A8B6=>8 P2A + 6 P2B
r21 [0 , 0]: A8B6C10=>8 P1A + 6 P1B + 10 P1C
r22 [0 , 0]: A8B6C10=>8 P2A + 6 P2B + 10 P2C
r23 [0 , 0]: A8B6C10D6=>8 P2A + 6 P2B + 10 P2C + 6 P2D
r24 [0 , 0]: B6C10=>6 P1B + 10 P1C
r25 [0 , 0]: B6C10=>6 P2B + 10 P2C
r26 [0 , 0]: B6C10D6=>6 P2B + 10 P2C + 6 P2D
r27 [0 , 0]: C10=>10 P1C
r28 [0 , 0]: C10=>10 P2C
r29 [0 , 0]: C10=>10 P3C
r30 [0 , 0]: C10D6=>10 P2C + 6 P2D
r31 [0 , 0]: C10D6=>10 P3C + 6 P3D
r32 [0 , 0]: D6=>6 P2D
r33 [0 , 0]: D6=>6 P3D
r34 [0 , 0]: C5D5=>5 P2C + 5 P2D
r35 [0 , 0]: C5D5=>5 P3C + 5 P3D

```

sources/ki3.in