

TDK-dolgozat

Szili László

Szili László
III. évfolyam

FOLYAMATSZINTÉZIS: KÖLCSONÖS KIZÁRÁSI FELTÉTELEK ALKALMAZÁSA KERESÉSI STRATÉGIÁKBAN

Témavezető: Dr. Bertók Botond

Műszaki Informatikai Kar, Rendszer- és Számítástudományi Tanszék

Pannon Egyetem
Intézményi Tudományos Diákköri Konferencia
2012



Tartalomjegyzék

1. Bevezetés	6
2. Folyamathálózat-szintézis	8
2.1. PNS feladatokról általánosságban	8
2.2. P-gráf módszertan	8
2.2.1. P-gráf reprezentáció	9
2.2.2. Kombinatorikusan lehetséges megoldásstruktúrák	9
2.2.3. A maximális struktúra	10
2.2.4. Döntési leképezések	10
2.2.5. SSG algoritmus	11
2.2.6. ABB algoritmus	12
3. Kölcsönös kizárás	13
3.1. A kölcsönös kizárás bevezetése	13
3.2. PNS kiterjesztése, és új jelölések	14
3.3. Egy létező megvalósítás kölcsönös kizárás kezelésére PNS-en belül	15
3.4. PNS hatékony kiterjesztése kölcsönös kizárásra	17
3.5. Matematikai Programozási modell kiterjesztése	19
3.6. Konklúzió a kiterjesztésről	20
4. A kölcsönös kizárás által biztosított gyorsítás kihasználása feladatmegoldásban	21
4.1. Játműhozzárendelési probléma	21
4.2. Játműhozzárendelési probléma mint PNS feladat	21
4.3. Kölcsönös kizárás értelmezése VAP feladatokban	22
4.4. Tesztelés szakirodalomból vett példák	23
4.4.1. 1. példa	23
4.4.2. 2. példa	25
4.5. Tesztelési eredmények kiértékelése	26
5. Hatványhalmaz generálás kölcsönösen kizáró részhalmazok esetén	27
5.1. Nagyságrendi különbségek megjelenése	27
5.2. Kizárási feltételnek eleget tevő részhalmazok számának becslése	27
5.3. Példa kölcsönös kizárást tartalmazó halmaz hatványhalmazának számosságváltozására	31
5.4. Új hatványhalmaz generáló eljárás	31
5.4.1. Motiváció	31
5.4.2. Az algoritmus bemutatása	32
5.5. Tesztelési eredmények	36
5.5.1. 1. példa	36
5.5.2. 2. példa	37
5.5.3. 3. példa	38
5.5.4. 4. példa	39
6. Továbblépési lehetőségek	41
7. Összefoglalás	41

8. Függelék	43
8.1. Az 1.feladat alap PNS modellje	43
8.2. Az 1.feladat kiterjesztett PNS modellje	44
8.3. Az 2.feladat kiterjesztett PNS modellje	45

Absztrakt

A folyamatszintézis sok területen megtalálható problémakör, legyen szó reakciósintézisről, ellátási lánc tervezésről vagy akár üzleti folyamatokról. Emellett bármilyen hasonló feladatkörben felismerhető, melyben a cél alapépítőkövek bizonyos kombinációjából egy teljes struktúra felépítése.

A feladatkör nehézsége a kombinatorikus robbanás jelenségből adódik, ugyanis az építőelemek számának növelésével exponenciálisan növekszik a lehetséges megoldásstruktúrák száma. Egyes feladatokban azonban a megfogalmazásból következően, esetleg bizonyos építőelemek tulajdonságaiból adódóan található olyan építőelem halmazok, melyek elemei egyszerre nem szerepelhetnek egy megoldásstruktúrában sem. Ezen kölcsönösen kizáró elemek miatt csökken a lehetséges megoldásstruktúrák száma, ami jelentős keresési tér és megoldási idő csökkenést is eredményez.

A megvalósítás azonban több okból kifolyólag sem triviális. A megoldó algoritmusok olyan módosítására van szükség, amely nem csak szűri a nem kívánt megoldásokat, hanem a szisztematikus keresés során eleve elkerüli azokat. Ehhez szükséges például egy új hatványhalmaz generáló eljárás, amely csak olyan részhalmazokat ad vissza eredményül, melyek eleget tesznek a kölcsönös kizárásoknak.

Munkám során kidolgoztam egy módszert, amely figyelembe veszi a kölcsönösen kizáró halmazok nyújtotta gyorsítási lehetőségeket, ugyanis a jelen ismert megoldási módok ezt nem teszik meg. A kidolgozott módszert megvalósítottam C++ programozási nyelven, valamint szakirodalomban elérhető példákra sikeresen teszteltem.

1. Bevezetés

Manapság a korlátozott erőforráskészletek, valamint a globálisan emelkedő energiaár miatt egyre nagyobb hangsúlyt kap az iparban különböző termelő folyamatok minél hatékonyabb megvalósítása. Ezek többnyire olyan folyamatok, melyekben előre definiált építőelemekből a kívánalmakat kielégítő, célnak legjobban megfelelő struktúra összeállítása a cél. Többnyire nyersanyagokból állítanak elő terméket. Az ilyen jellegű problémákat folyamat hálózat szintézisnek nevezzük (Process-Network Synthesis - PNS). A folyamatszintézis sok területen megtalálható problémakör, legyen szó reakciósintézisről, ellátási lánc tervezésről vagy akár üzleti folyamatokról. A feladat egyértelmű reprezentálására, és megoldására Friedler és munkatársai az 1990-es években dolgoztak ki egy széleskörűen alkalmazható keretrendszert. Mely képes a problémák ábrázolására, valamint kombinatorikus algoritmusokon keresztül a optimális megoldásának, sőt akár az n -legjobb megoldásának meghatározására is. Munkám során ezt a módszert vizsgáltam, és bővítettem ki olyan gyorsítási lehetőségekkel, melyek a feladatok struktúrális jellegéből adódnak. Az általam vizsgált ilyen tulajdonság a kölcsönösen kizáró műveleti egységek megjelenése volt, mint korlátozási tényező. Dolgozatomat az alábbi logikai szisztéma szerint állítottam össze.

A második fejezetben rövid áttekintést nyújtok magáról a PNS problémáról, és annak meghatározó jellemzőiről. Ezt követően röviden bemutatom a kezelésére kifejlesztett P-gráf módszertant. Mely magába foglalja a feladatosztály P-gráf reprezentációját, struktúrális tulajdonságait, valamint a megoldásra kifejlesztett algoritmusokat. Ezen keresztül megismerjük azokat az alapmódszereket, melyekre alapostam a további kutatásomat.

A harmadik fejezetben térek ki az általam kiemelt részre, mely a kölcsönös kizárás bevezetése. Itt bemutatom, hogy mit is jelent a kölcsönös kizárás megjelenése PNS problémákban, hogyan tudjuk ezt modellezni, és milyen megoldási módok állnak jelenleg az ilyen problémák kezelésére. Ezt követően bevezetem a PNS probléma kiterjesztését kölcsönösen kizáró halmazok kezelésére. Majd bemutatom azokat a módosításokat a megoldó algoritmusokban, melyek következtében azok megoldás során képesek lesznek struktúrális jelleg alapján kiszűrni a kölcsönös kizárási feltételnek nem eleget tevő részproblémákat. A nehézség abban jelentkezett, hogy ez az irányvonala a PNS problémának viszonylag friss kutatások eredménye, így nem áll róla elegendő leírás a szakirodalomban, ezen túl bizonyos részei a saját eredményeimet is tükrözik.

A negyedik fejezetben illusztrálom a kölcsönös kizárás figyelembevételének feladatmegoldás során fellépő hatékonyságát, szakirodalomból vett járműhozzárendelési példákra. Ehez segítségül röviden bemutatom, hogy mi is az a járműhozzárendelési probléma. Ezt követően azt, hogy hogyan tekinthetünk a problémára, mint PNS probléma, valamint, hogy minek köszönhetően értelmezhetjük a kölcsönös kizárást a struktúrában. Még szintén ebben a fejezetben összehasonlítom a kölcsönös kizárással bővített PNS modellt és az eredeti PNS modell feladatmegoldásának hatékonyságát járműhozzárendelési feladatokra.

Az ötödik fejezetben bemutatom, saját kutatási eredményeimet. Munkám során a kölcsönös kizáráshoz kapcsolódóan kifejlesztettem egy olyan hatványhalmaz generáló algoritmust, mely képes figyelembevenni az alaphalmazban megjelenő kölcsönösen kizáró részhalmazokat, és szisztematikusan csak azokat

a lehetséges részhalmazokat generálja le, melyek ezen kölcsönösen kizáró feltételeknek eleget tesznek. Az algoritmust implementáltam C++ programozási nyelven, így bemutatom a kapott teszteredményeimet is, és összehasonlítom azokat az eddig ismert kevésbé hatékony módszerrel.

Munkám motivációját az adta, hogy az általam kidolgozott hatványhalmaz generáló eljárás beépíthető a P-gráf keretrendszer megoldó algoritmusába, így a későbbiekben még nagyobb hatékonyságnövelés is elérhető kölcsönös kizárást tartalmazó feladatok megoldásában. Jövőbeli célom ennek a megvalósítása.

2. Folyamathálózat-szintézis

Ebben a fejezetben, a téma szakirodalmára hivatkozva, bemutatom a folyamat hálózat-szintézis feladatkört, majd ezt követően a megoldására kidolgozott P-gráf módszertant. Mivel a rendelkezésre álló szakirodalom rendkívül terjedelmes, így a jelen fejezetben csak az olyan fontos dolgokat szeretném bemutatni, melyek hiánya befolyásolná a dolgozatban foglalt feladatkiterjesztések érthetőségét. Így a részletes tárgyalás helyett egy alapot szeretnék biztosítani a későbbi eredmények érthetőségére. Mindezt annak reprezentálásaként, hogy honnan indult az alap kutatás, melyet az én témám is alapjául vesz.

2.1. PNS feladatokról általánosságban

A folyamathálózat-szintézis ipari alkalmazásokban, gyakran felmerülő problémakör. Melynek hatékony megoldása egyre nagyobb figyelem kap a nökvő nyersanyag és energiaárak miatt. A feladatkör többnyire termelő folyamatokat foglal magába. A feladat célja előre ismert alapanyagokból előre definiált termékek gyártása oly módon, hogy az előre ismert, és rendelkezésre álló építőelemekből valamilyen szempontból optimális hálózatot meghatározása, mely képes a kívánt termékek leggyártására.

Az építőelemeket jelen esetben műveleti egységeknek nevezzük, bemeneteit és kimeneteit pedig anyagoknak. A komplett hálózat műveleti egységek ki és bemeneti anyagainak összekapcsolásával állítható össze. Ezáltal alapvetően egy kombinatorikus optimalizálási problémaként kezelhetjük. Azonban a műveleti egységekhez rendelhető költségfüggvények és állapotváltozók miatt egy vegyes egész típusú matematikai programozási feladatként is tekinthetünk rá egyben. Kettős természete miatt az optimum meghatározása nehéz, és rendkívül számításgényes feladat. A feladat nehézségét az is mutatja, hogy több heurisztikus módszert is kidolgoztak a gyorsabb megoldás érdekében, azonban ezek nem tudják garantálni a megoldás optimalitását. Ezek többnyire nem használják ki a PNS kombinatorikus jellegét, így csak a matematikai modell alapján dolgoznak. A továbbiakban egy olyan módszertant szeretnék bemutatni röviden, mely rendelkezik a problémakör modellezéséhez egyértelmű struktúrareprezentációval, valamint bizonyítottan optimális megoldást adó megoldómódszerrel is. Ez nem más, mint a Friedler és munkatársai által kidolgozott P-gráf módszertan.

Egy PNS feladat struktúrális modelljét egy (P, R, O) hármassal definiálhatjuk. Emellett feltételezzük, hogy létezik egy véges halmaza a megjelenő anyagoknak, jelöljük ezt M el. Ezt követően P jelöli az előállítani kívánt anyagok halmazát ahol $P \subset M$ valamint $P \neq \emptyset$, R jelöli a rendelkezésre álló nyersanyagok halmazát, melyekre igaz, hogy:

$$R \subset M, P \neq \emptyset \text{ valamint } R \cap P = \emptyset.$$

Továbbá O jelöli a rendelkezésre álló műveleti egységek véges halmazát.

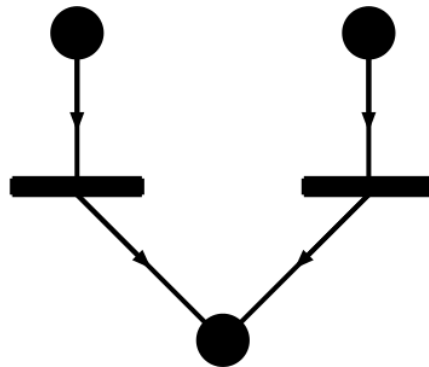
2.2. P-gráf módszertan

A PNS probléma általános bemutatását követően kitérek a P-gráf módszertan rövid bemutatására. Csak a fontos, és a későbbiek folyamán szükséges része-

ken haladok végig. A módszertan teljes leírásáról bővebb információ található [3],[4],[5],[6]-ben.

2.2.1. P-gráf reprezentáció

A PNS feladat egyértelmű reprezentálásához speciális ábrázolásmód szükséges. Hagyományos gráfon nem lenne egyértelmű az ábrázolás, mivel kétfajta csúcshalmazzal rendelkezik a PNS probléma, melyek az anyag és műveleti egység típusúak. Ezáltal ez a két csúcshalmaz megkülönböztetésére van szükség. Az egyértelmű jelölés bevezetése érdekében az eredeti szerzők bevezették a folyamatgráfot (Process Graph - P-gráf). A P-gráf egy irányított páros gráf, melynek két különböző típusú csúcsa van. Az O típusú csúcsok reprezentálják a műveleti egységeket, és jelölésük fekete téglalappal történik, valamint az M típusú csúcsok jelentik a megjelenő anyagokat, reprezentációjuk telített karikával történik. Egy ilyen P-gráf részlet látható az 1. ábrán. Az irányított élek reprezentálják



1. ábra. P-gráf reprezentáció

az anyagáramokat, a nyíl iránya megegyezik az anyag áramlásának irányával. Matematikailag megfogalmazva a P-gráfot egy (m, o) kettesként definiálhatjuk, ahol m halmazba az M típusú csúcsok, az o halmazba pedig az O típusú csúcsok tartoznak.

2.2.2. Kombinatorikusan lehetséges megoldásstruktúrák

Mielőtt bevezetnénk a kombinatorikusan lehetséges struktúra fogalmát, először definiáljuk, mit is értsünk lehetséges struktúra alatt. Ennek a meghatározására a következő definíciót alkották meg. Egy struktúra akkor lehetséges, ha minden $o \in O$ műveleti egységről eldönthető, hogy benne van-e az adott struktúrában, vagy sem. Ez a meghatározás azonban nem kielégítő. Az összes lehetséges struktúra halmazából csak nagyon kevésre igaz, hogy a gyakorlatban megvalósítható rendszert leíró struktúra. Ezt követően bevezették a kombinatorikusan lehetséges struktúra fogalmát, mely olyan struktúrákat foglal magába, melyek a gyakorlatban megvalósíthatóak.

Van néhány olyan struktúrális vagy kombinatorikus tulajdonság, melyel minden gyakorlatban megvalósítható struktúrának rendelkeznie kell. Ezeket a tulajdonságokat Friedler és társai mint axiómákat vezették be, melyek:

- (S1) Minden végtermék szerepel a struktúrában. $P \subseteq m$
- (S2) Egy, a struktúrában szereplő anyag akkor és csak akkor nyersanyag, ha a struktúrában szereplő műveleti egységek közül egyetlen sem állítja elő.
- (S3) A struktúrában szereplő összes műveleti egység a feladatban előre definiált.
- (S4) A struktúrában szereplő összes műveleti egységből vezet út legalább egy végtermékbe.
- (S5) A struktúrában szereplő összes anyag valamely műveleti egységnek vagy kimenete vagy bemenete.

Ezt követően akkor mondjuk, hogy egy megoldás kombinatorikusan lehetsége, ha az kielégíti az (S1)-(S5) axiómákat. Egy kombinatorikusan lehetséges struktúráról nem mondhatjuk ki, hogy biztosan valós megoldás is lesz, mivel lehet, hogy nem a kellő mennyiségben állítja elő a termékeket. Tehát az axiómák szükségesek, de nem elégséges feltételt adnak a megvalósítható megoldásokra.

2.2.3. A maximális struktúra

Abból kiindulva, hogy megoldásstruktúrákkal is végezhetőek műveletek megvizsgálták, és bizonyították, hogy ezek további tulajdonságokkal rendelkeznek. Zártak az unió képzésre, kompozíciójuk állítható elő, vagy akár dekomponálhatók. Emellett bizonyították, hogy egy megoldásstruktúrát bővítve szintén megoldásstruktúrát kapunk, így azok bővíthetők. Mivel a megoldásstruktúrák kiterjeszthetők, és zártak az unióra, léteznie kell egy olyan megoldásstruktúrának, amely tartalmazza az összes lehetséges megoldásstruktúrát. Ezt a struktúrát a feladat maximális struktúrájának nevezték el.

Az MSG algoritmus

A maximális struktúra meghatározására az MSG algoritmust alkották meg, mely polinomidőben határozza meg azt a fent említett 5 axióma alapján. Az algoritmus működése két fő részre osztható. Az első redukciós szakaszban az algoritmus kiszűri azokat az anyagokat és műveleti egységeket amelyek megsértik az axiómákat, és így egyik megoldásstruktúrába sem kerülhetnek bele. Ezt követően a második részben szintén az axiómákra alapozva építi fel az algoritmus a maximális struktúrát az első rész futása utána megmaradt műveleti egységekből, és anyagokból. Maga az algoritmus szisztematikus, és bizonyítottan helyes lépésekből áll. A maximális struktúra jelentősége abból adódik, hogy ez a struktúra fogja kiindulási alapját képezni a megoldó algoritmusoknak.

2.2.4. Döntési leképezések

Vezessünk be egy leképezést, Δ -t, ami minden anyaghoz megadja azokat a műveleti egységeket, melyeknek az adott anyag kimeneti anyaga. Formálisan $\Delta(X) = \varphi^-(X)$ ahol $X \in m$, ha (m, o) az (P, R, O) PNS feladat P-gráfja.

Ezt követően legyen $\delta[m]$ döntési leképezés olyan párok $(X, \delta(X))$, $X \in m$ halmaza, melyek első tagja egy anyag, második tagja pedig egy halmaz melybe

olyan műveleti egységek tartoznak, amelyek ezt az anyagot gyártják, vagyis egyes anyagokhoz rendeljük hozzá az őket előállító műveleti egységek egy halmazát. A könnyebb érthetőség kedvéért megjegyezzük, hogy $\delta()$ jelölés egy anyagra, míg $\delta[]$ jelölés anyagok egy halmazára vonatkozik. A döntési leképezésekről bővebb információt találhatunk [5].

A döntési leképezések konzisztenciája elengedhetetlen a PNS problémák megoldásánál. A kombinatorikusan lehetséges struktúrák generálásához alternatív döntések vagy döntések sorozatát szükséges végrehajtani. A döntések műveleti egységek beveteléről vagy kizárásáról történnek, és ezek a döntések a döntési leképezések segítségével írhatóak le. Minden terméket legalább egy műveleti egységnek kell gyártani, továbbá legalább egy műveleti egységnek kell gyártani egy olyan anyagot, amelyet egy korábbi döntés során bevett műveleti egység fogyaszt. Az előző eset alól kivételt képeznek a nyersanyagok. A döntések során vigyázni kell arra, hogy a már kizárt műveleti egységeket egy másik döntés során már nem lehet beválasztani, ez inkonzisztenciát okozna.

Az inkonzisztens döntések elkerülése érdekében a műveleti egységeket három osztályba kell sorolni, melyek a következők: az adott struktúrába beválasztott műveleti egységek halmazába *inc*, az adott struktúrából kizárt műveleti egységek halmazába *exc*, illetve azoknak a műveleti egységeknek a halmazába, amelyekről még nem született döntés.

A következőkben vizsgáljuk meg, hogy mi alapján működik az SSG és az ABB algoritmus, és hogyan segíti a működésüket a döntési leképezések bevezetése.

Először is nevezzük aktív halmaznak anyagok azon halmazát, melyek előállításáról dönteni kell. Nyilvánvalóan ez a halmaz kiindulási esetben a termékeket tartalmazza. Egy döntés során az anyagot előállítani képes műveleti egységek közül azokat kell meghatározni, amelyek az adott struktúrában előállítják az anyagot. Az így kiválasztott műveleti egységek bekerülnek a struktúrába, a többi pedig ki kell zárni. Ezt követően minden döntés után frissül az aktív halmaz. Azok az anyagok, melyek gyártásáról már döntöttünk, kikerülnek belőle, és olyan nem nyersanyag anyagok kerülnek be, melyeket egy már beválasztott nyersanyag fogyaszt, és még nem volt rajtuk döntés. Abban az esetben, ha az aktív halmaz üressé válik a döntések során, a bevett halmaz elemei fogják alkotni a lehetséges struktúrát.

2.2.5. SSG algoritmus

Az (Solution Structure Generator - SSG) algoritmus minden megoldásstruktúrát előállít a döntések összes lehetséges sorozatának leszámolásával, még hozzá bizonyítottan csak egyszer. Az algoritmus döntési leképezéseket alkalmazó rekurzív eljáráshíváson alapszik. A rekurzió addig tart, míg a fentebb említett aktív anyaghalmaz üres nem lesz minden esetben. Ez a folyamat úgy ábrázolható, mint egy olyan fagráf bejárása, melyben a csúcsok az anyagokon végzett döntéseket, a csúcsból kiinduló élek pedig a lehetséges döntési alternatívákat reprezentálják. A kombinatorikusan lehetséges megoldások ennek a fagráfnak a levelein fognak elhelyezkedni.

2.2.6. ABB algoritmus

A feladat optimális megoldását úgy is megkaphatnánk, hogy először SSG algoritmussal generáljuk az összes megoldást, majd mindegyikre kiszámoljuk a célfüggvény szerinti értéket, majd ez alapján kiválasztjuk közülük a legkisebb értékhez tartozó megoldást. Ez a módszer azonban nem hatékony abban az esetben, ha csak az optimális megoldásra van szükségünk.

Az ABB algoritmus a valamilyen szempontból optimális megoldás megtalálására lett kifejlesztve. Maga az algoritmus az SSG analógiájára alapul, azonban a különbség annyi, hogy döntés előtt az algoritmus egy korlátszámítási eljárást hajt végre, amely a részproblémákhoz egy alsó korlátot rendel. Ez az alsó korlát alapján elvethető részproblémákat ismer fel, így az ABB eljárás az SSG által bejárt fa csak egy részét generálja ki. Ezen a kigenerált részfán halad folyamatosan tovább mindaddig míg el nem éri a levelet. Az algoritmus annál hatékonyabb, minél kisebb a bejárt fa, amit döntően a korlátozási eljárásban meghatározott alsó korlát élessége határoz meg.

3. Kölcsönös kizárás

3.1. A kölcsönös kizárás bevezetése

Mint azt az irodalmi áttekintés során láthattuk a folyamatszintézis sok területen megtalálható problémakör, legyen szó reakciósintézisről, ellátási lánc tervezésről vagy akár üzleti folyamatokról. Egyes feladatokban azonban a megfogalmazásból következően, esetleg bizonyos építőelemek tulajdonságaiból adódóan találhatók olyan építőelem halmazok, melyek elemei egyszerre nem szerepelhetnek egy megoldásstruktúrában sem. A következőben nézzünk pár ilyen példát, mely problémakörben jelenhetnek meg kölcsönösen kizáró halmazok.

Gyártórendszerek tervezésekor megjelenő probléma lehet, hogy bizonyos műveleti egységek olyan formájúak, vagy akkora helyet foglalnak el, hogy egyszerre a kiválasztott darabok nem férnek el egymás mellett a kívánt helyen. Továbbá technológiai megfontolásból a tervezés során több lehetséges kivitelezhető technológiát vizsgálunk, ám azok egyszerre történő alkalmazása nem megoldható, nem költséghatékony, esetleg mindegyik nyersanyagellátása nem biztosítható egyszerre. Például egy pékségben egyszerre nem alkalmaznának villany és gázfűtésű kemencéket is. Valamint olyan esetek is előfordulhatnak, hogy bizonyos berendezések olyan mellékterméket állítanak elő, melyek jelentősen képesek károsítani egyes berendezéseket, melyek a folyamat későbbi szakaszán foglalnának helyet. Ebben az esetben sem célszerű ezeket egyszerre szerepeltetni a megoldásstruktúrában, így definiálhatjuk őket kölcsönösen kizáróként.

Mindezek mellett még egy fontos területet említenék, ahol megjelenik kölcsönös kizárás olyan helyen, amely kötődik a folyamatszintézishez. Ez nem más, mint az üzleti folyamat szintézis. Manapság kulcsfontosságú szerepet tölt be egy vállalat számára az a tény, hogy versenyképes legyen. Ennek tükrében minél pontosabban és körültekintőbben próbálják tervezni azt a stratégiát, mely szerint fognak haladni, és emellett nagy ügyet fordítanak alternatív lehetőségek vizsgálatára is. Azonban az üzleti folyamatok között gyakran megjelennek olyan döntések, melyek kizáró vagy műveletet eredményeznek a lehetséges további tevékenységek között, így azok közül csak egyet lehet végrehajtani. Egy egyszerű példaként említhetünk egy befektetés elfogadását, vagy elutasítását. Ebben az esetben sem teljesülhet egyszerre mindkettő. Ám az ilyen kölcsönösen kizáró esetnek a modellezése, és alternatív megoldás generálása nem teljesen kiforrott.

További motiváció a bevezetésre

A kölcsönösen kizáró halmazok bevezetésével új korlátozásokat kapunk a lehetséges megoldásokra. Minden egyes új korlát bevezetése egy adott problémához csökkentheti a lehetséges megoldások, és a megoldás során generált részproblémák számát. Valamint a korlátoknak köszönhetően az egész keresési tér csökkenhet, amit a feladat megoldása során be kell járni, akár jelentős mértékben. A keresési tér csökkentés kihatással lenne az adott probléma megoldási idejére, akár rendkívüli módon meggyorsíthatja azt. Ennek tükrében a kölcsönösen kizáró halmazok bevezetésére olyan formában van szükség, melyen keresztül azokat új korlátozásként vezethetjük be a modellbe, és emellé olyan szisztematikus megoldómódszert tudunk alkalmazni, amely már a megoldás meghatározása

közben figyelmen kívül hagyja az olyan részmegoldásokat, melyek nem tesznek eleget a kölcsönös kizárásnak. Ha mindez megvalósítható, akkor azzal nagymértékben gyorsíthatjuk a jelenleg elérhető megoldási módokat.

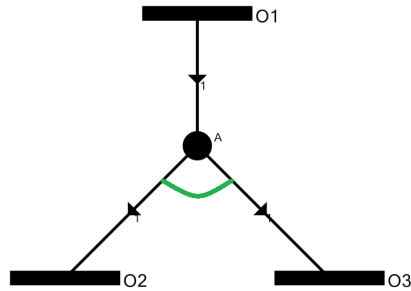
3.2. PNS kiterjesztése, és új jelölések

Mint azt láthattuk, több területen is szükség van a kölcsönös kizárás kiterjesztésére a feladatkör pontosabban történő leírásából adódóan is. Továbbá ismeretes a P-gráf keretrendszer folyamatszintézis problémák modellezésére, hatékony megoldására, valamint akár alternatív megoldások, vagy akár az első n -legjobb megoldás meghatározására. Mindemiattn célszerűnek tartottuk a keretrendszer képessé tételét kölcsönösen kizáró halmazok figyelembevételére, ezzel tovább növelve alkalmazhatóságának körét. Azért választottuk ezt, mert rendelkezik már az előre feltételezett ábrázolási, és szisztematikus keresési tulajdonságokkal, csak képessé kell tenni a módszert, és kiterjeszteni a kölcsönösen kizáró halmazok kezelésére.

Vezessünk be új jelölést a kölcsönösen kizáró halmazok ábrázolására. Legyen OX az a halmaz, amely tartalmazni fogja az adott feladatban definiált j darab kölcsönösen kizáró halmazt ($ox_1 \dots ox_j$).

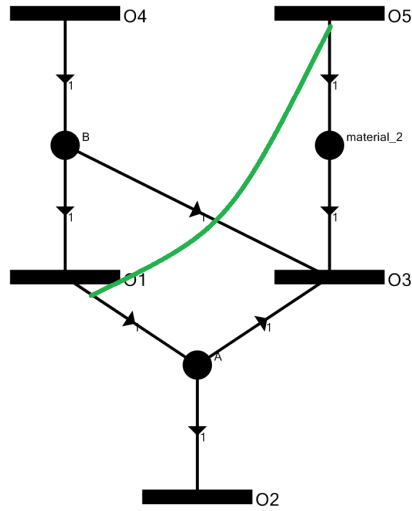
$$OX = \{ox_1, \dots, ox_j\}$$

Továbbá szükség lesz új grafikus ábrázolásra is, mellyel egy P-gráfon szemléltetni tudjuk a kölcsönösen kizáró műveleti egységeket. Vezessük be erre a célra a következő jelölést. Az egy anyagcsúcsba beérkező, vagy egy anyagcsúcsból kiinduló kölcsönösen kizáró műveleti egységhez tartó éleket kössük össze egy ívvel ahogyan az a 2.ábrán is látható. Itt azt láthatjuk, hogy O2 és O3 műveleti egységek kölcsönösen kizárók. Vagy olyan esetben, ha a kölcsönösen kizáró



2. ábra. Új jelölés kölcsönösen kizáró műveleti egységek reprezentálására

műveleti egységeknek nincs közös bemenő, vagy kimenő anyaga, akkor kössük őket össze egy görbével, ahogy az a 3.ábrán látható. Ezen az ábrán O1 és O5 műveleti egységek kölcsönösen kizáróak. A jelölések kiegészítését követően a feladat pontosabb definiálását egy lehetséges megvalósítás után fogom részletesen bemutatni.

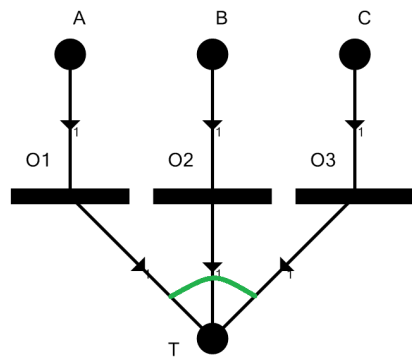


3. ábra. Új jelölés kölcsönösen kizáró műveleti egységek reprezentálására

3.3. Egy létező megvalósítás kölcsönös kizárás kezelésére PNS-en belül

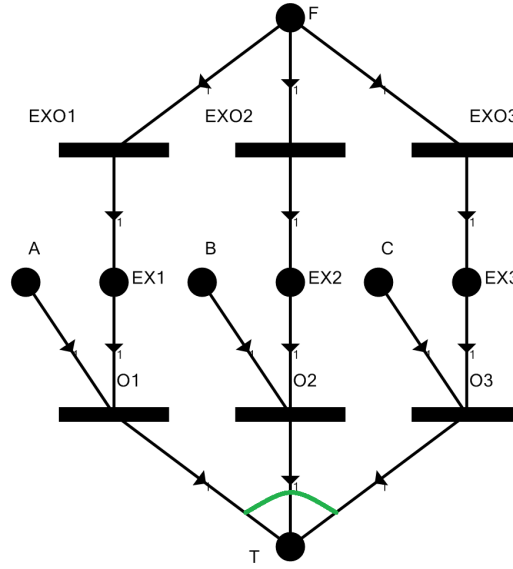
Alapesetben a PNS modellben nincs olyan megkötés, amely definiálná azt, hogy egy anyagot egyszerre maximum csak egy arra képes műveleti egység állíthat elő. Azonban született erre a célra már olyan modellfelírás, amely segítségével valamilyen szinten kezelni lehet ezt a fajta problémát is. Ez az alap modell trükkös kibővítésével kezeli a problémát.

Röviden a módszer lényege annyi, hogy segéd nyersanyagok, segéd műveleti egységek, és segéd köztes anyagok kerülnek beiktatásra a modellbe, majd ezek alsó és felső korlátainak megfelelő beállításával elérhető a kívánt kölcsönös kizárás. Ezt szeretném bemutatni most az 4.ábrán látható P-gráf részleten.



4. ábra. P-gráf részlet

Az ábrán 3 nyersanyagot (A, B, C), 3 kölcsönösen kizáró műveleti egységet O_1, O_2, O_3 , valamint 1 terméket T láthatunk. Jelen esetben T terméket szeretnénk előállítani, melyre 3 különböző módszert ismerünk, ám a 3 módszer közül választanunk kell, és csak egyet használva állíthatjuk elő T -t. Ezt a különböző 3 módszert reprezentálják O_1, O_2, O_3 műveleti egységek, melyek külön nyersanyagokat felhasználva (A -t, B -t és C -t) képesek előállítani T -t. Első lépésben adjunk hozzá a modellhez egy kiinduló nyersanyagot (F), 3 új műveleti egységet EXO_1, EXO_2, EXO_3 , valamint 3 új köztes terméket EX_1, EX_2, EX_3 . A három új hozzáadott műveleti egység bemenete az újonnan hozzáadott nyersanyag, kimenetei pedig az újonnan hozzáadott új köztes termékek lesznek. Valamint az eddigi műveleti egységek bemenetein meg kell jelentetni a hozzáadott köztes termékeket. Az új modell P-gráf ábrázolása látható a 5. ábrán.



5. ábra. A kiegészített P-gráf részlet

Ez a kiterjesztés azonban még nem elég a kölcsönös kizárás teljes mértékű modellezésére. Új műveleti egységek alsó kapacitáskorlátját állítsuk 1-re. Ebben az esetben a bevezetett új nyersanyagból az adott példában legalább 3-nak rendelkezésre kell állni, hogy minden műveleti egység egyszerre működhessen. Azonban ha ezt 3-tól kisebb értékre állítjuk, akkor nem lesz elég bemenő nyersanyaga mindhárom új műveleti egységnek, így azok közül nem szerepelhet mindhárom egyszerre. Megjegyzendő az, hogy egy műveleti egység akkor működőképes, ha az összes szükséges bemenete elérhető a struktúrában. Továbbá vegyük észre azt, ha a bevezetett nyersanyag mennyiségét maximalizáljuk 1-re, és a hozzáadott műveleti egységek kapacitását pedig minimalizáljuk 1-re, akkor olyan eset áll fent, hogy a bevezetett 3 új műveleti egység közül mindig pontosan csak az egyik fog tudni működni, mivel csak 1 bemenő anyagának kielégítésére elegendő a megadott 1 felső korlát. Így EO_1, EO_2, EO_3 műveleti egységek önálló működési feltétele a nyersanyag mennyiségének tekintetében ellehetetleníti a másik kettő működését.

3.4. PNS hatékony kiterjesztése kölcsönös kizárásra

Az imént említett megoldás jó megoldás ugyan, azonban rendelkezik egy-
rendkívüli hátránnyal. Mint az ismeretes egy PNS feladat megoldása során a
modellben szereplő műveleti egységek száma kulcsszerepet játszik, mivel a meg-
oldási időt jelentős mértékben befolyásolja. Akár exponenciálisan növekedik a
műveleti egységek számának növekedésével. Emiatt a kölcsönös kizárás repre-
zentálása végett hozzáadott műveleti egységek a megoldási időt rendkívüli mó-
don megnövelhetik, valamint a generált modellt is rendkívül megbonyolíthatják,
ezzel akár ellehetetlenítve egy nagyobb példa belátható időn belül történő meg-
oldását. Ebből kifolyólag egy olyan módszerre volt szükség, amely bevezetése
nem befolyásolja a generált modell bonyolultságát, valamint a benne megjelenő
műveleti egységek számát.

A PNS modelljét olyan módon szeretnénk módosítani, hogy az képes legyen
kölcsönösen kizáró halmazok hatékony kezelésére, valamint a feladat megoldása
során az eddig elérhető algoritmusok működése, és hatékonysága legalább ugyan-
olyan jó legyen, mint eddig. A továbbiakban ez alapján szükség lesz majd az
algoritmusok módosítására is a kölcsönös kizárás figyelembevétele érdekében,
így induljunk el az alapoktól.

Mivel a modell kiterjesztéséről beszélünk, így természetesen a modell, vala-
mint a megoldó algoritmusok működésének nem szabad eltérő működést pro-
dukálnia, ha nem tartalmaz az adott probléma kölcsönös kizárását. Ebben az
esetben úgy kell működnie, mint eredetileg. Első lépésként vezessünk be halmaz
reprezentációt, a kölcsönösen kizáró halmazok kezelésére. Mint azt korábban be-
vezettem jelölje a kölcsönösen kizáró halmazok halmazát OX , mely tartalmazza
a feladatban rögzített j darab kölcsönösen kizáró halmazt.

$$OX = \{ox_1, \dots, ox_j\}$$

Mivel olyan halmazokról beszélünk, melyek elemei kölcsönösen kizáróak, így
egy további feltevésünk lehet velük kapcsolatban. Egy halmazban csak akkor
lehet kölcsönös kizárásról, beszélni, ha annak valóban van annyi eleme, hogy
tudják egymást kölcsönösen kizárni. Minden ilyen halmazról feltételezhetjük a
továbbiakban, hogy számosságuk nagyobb mint 1.

$$\forall ox \in OX \text{ teljesül, hogy } |ox| > 1$$

Ezt követően a problémát máris felírhatjuk mint (P, R, O, OX) , ahol a (P, R, O)
PNS feladatra értelmeztük a kölcsönösen kizáró műveleti egység halmazokat. A
jelölések az irodalmi áttekintésben bevezetettnek tesznek eleget, vagyis továbbra
is R jelöli a nyersanyagokat, P a legyártandó termékeket, valamint O a rendel-
kezésre álló műveleti egységeket. Emellett itt is érvényes az, hogy véges elemű
 M anyaghalmazt feltételezünk. Egy további megkötésünk még kell, hogy legyen
 OX elemeire nézve, még hozzá az, hogy csak olyan elemeket tartalmazhatnak,
melyek az alapfeladatban is definiálva vannak. Vagyis minden $ox \in OX$ hal-
mazra igaznak kell lennie, hogy $ox \subseteq O$.

$$\forall ox \in OX \text{ teljesül, hogy } ox \subseteq O$$

A kölcsönös kizárás minden esetben kétoldalúan működik, tehát nem for-
dulhat olyan elő, hogy egy műveleti egység kizár egy másikat, azonban az nem
zárja ki az őt kizárót.

Vezessünk be egy olyan leképezést ez alapján a döntési leképezések mintájára, mely egy műveleti egységhez hozzárendeli a vele kölcsönösen kizáró műveleti egységeket, valamint egy nem kölcsönösen kizáró műveleti egység halmazhoz hozzárendeli a halmaz elemek által kizárt összes műveleti egységet. Jelölje ezt a leképezést $\tau(m)$, ha halmazra használjuk akkor pedig a döntési leképezések alapján $()$ helyett $[]$ -t használunk. Formálisan:

$$\tau(m) = \left\{ \bigcup_{ox \in OX} ox, \forall ox\text{-re, amire fennáll, hogy } m \in ox \right\} \setminus m$$

$$\tau[m] = \left\{ \bigcup_{ox \in OX} ox, \forall ox\text{-re, amire fennáll, hogy } m \subseteq ox \right\} \setminus m$$

A formális leírást követően vizsgáljuk meg milyen strukturális változások, és megoldás közbeni különbségek léphetnek fel a kölcsönös kizárás bevezetésével. Kezdjük a kombinatorikusan lehetséges megoldásstrukturák meghatározásával. Mivel a modell kiterjesztéséről van szó, így (S1)... (S5) axiómahalmaznak továbbra is érvényesnek kell lennie. Azonban minden megoldásstrukturától elvárjuk, hogy teljesítse a kölcsönös kizárást, így egy további axiómát szükséges bevezetni, melyet jelöljünk (S6)-al.

(S6) Egy megoldásstruktúra nem tartalmazhat kölcsönösen kizáró részhalmazt.

$$\forall ox \in OX : \text{igaz, hogy } |o \cap ox| \leq 1$$

Ahol o az a halmaz, mely az adott megoldásstruktúrában szereplő műveleti egységeket tartalmazza. A bevezetett (S6) axióma azt mondja ki, hogy egy megoldásstruktúrában egyszerre egy kölcsönösen kizáró halmazból egyszerre maximum csak egy elem lehet jelen. Ez a feltétel biztosítja azt, hogy teljesülni fog a kölcsönös kizárásra tett feltételünk. Így akkor mondjuk, hogy egy struktúra kombinatorikusan lehetséges, ha teljesíti az (S1)... (S6) axiómák mindegyikét.

Az eredeti megoldó algoritmusok nagymértékben a döntési leképezésekre alapulnak. Ennek következtében a kölcsönösen kizáró halmazokat a döntési leképezésekkel is együtt kell tudni használni, és vigyázni kell arra, hogy a döntések továbbra is egymással konzisztensek legyenek. A megoldás során a műveleti egységeket három osztályba szokták sorolni, melyek a következők:

- *inc* : az adott struktúrába már bevásárolt műveleti egységek halmaza.
- *exc* : az adott struktúrából kizárt műveleti egységek halmaza.
- *p* : olyan műveleti egységek halmaza, mely szerepel az adott struktúrában, de még nem született döntés róla. $p = (o \setminus inc) \setminus exc$

Egy döntés során mindig arról döntünk, hogy az m anyaghalmazt előállítani képes műveleti egységekből melyek kerülnek be az adott struktúrába, és melyeket fogjuk kizárni. A kölcsönösen kizáró halmazok bevezetésével azonban ez módosul, ugyanis nem csak azokat a műveleti egységeket zárhatjuk ki, melyeket nem választottunk ki m anyaghalmaz előállítására, hanem azokat is ki kell zárunk, melyek a bevett elemek *inc* halmazának elemeivel kölcsönösen kizáróak. Példaként levezetve: jelölje x azt a részhalmazt, melyről úgy döntöttünk,

hogy a jelen esetben beveszünk inc -be, mert előállítja m halmaz elemeit. (m halmazt előállító műveleti egységek halmazát visszaadó leképezést jelöli $\varphi^-[m]$) Továbbá csak olyan műveleti egységekről szeretnénk dönteni, amikről eddig még nem döntöttünk. Tehát:

$$x \subseteq (\varphi^-[m] \cap p)$$

$$inc = inc \cup x$$

Ekkor a kizárt halmazhoz nem csak azokat az elemeket kell hozzáadni, amikről eldöntöttük, hogy nem szerepelhetnek az adott m anyaghalmaz előállításában, hanem azokat is ki kell zárni, melyeket a hozzáadott halmaz elemei kizárnak.

$$exc = exc \cup ((\varphi^-[m] \cap p) \setminus x) \cup \tau[x]$$

Ezt követően csökkenteni kell azoknak a műveleti egységeknek a halmazát, amelyekről még nem született döntés.

$$p = (p \setminus inc) \setminus exc$$

Most láthatjuk, hogy miért volt érdemes bevezetni a $\tau(m)$ leképezést, és hogyan tudtuk felhasználni arra, hogy az eddig meglevő algoritmusok kezelni tudják a kölcsönösen kizáró halmazokat a döntési leképezések segítségével. Ennek köszönhetően minden olyan eddigi algoritmus ki tudja használni a kölcsönösen kizáró halmazok figyelembe vételét, amely a döntési leképezésekre alapul (például SSG, ABB).

3.5. Matematikai Programozási modell kiterjesztése

Mint azt az irodalmi áttekintésben említettem, az ABB algoritmus szintén döntési leképezéseken alapul. Azonban a döntés meghozatala előtt elvégez egy korlátszámítási eljárást, mely egy alsó korlátot rendel minden részproblémához, és azok ez az érték alapján elvethetőek, vagy megtarthatóak a megoldás további keresésének szempontjából. Ez a korlátszámítás egy matematikai programozási modell megoldását foglalja magába. A kölcsönös kizárás hatékony működése, valamint a korlátok további validitása miatt képessé kell tenni ezt a modellt is a kölcsönösen kizáró halmazokkal történő számolásra. A modell pontosabban egy vegyes egész típusú programozási modell lesz (Mixed Integer Linear Programming - MILP). A modell megoldási módjának részletes bemutatásával a dolgozat keretein belül nem foglalkozom, csak annak módosításával. Egyébként többnyire az egész értékű változók relaxációjával történik a megoldása.

Továbbá y_j legyen egy bináris változó, mely 1 vagy 0 értéket vehet csak fel abban az esetben, ha o_j műveleti egység benne lesz az adott megoldás struktúrában, vagy nem. Mivel szerepelnek kölcsönösen kizáró halmazok, melyek mindegyikéből csak egyetlen elem szerepelhet egyszerre a megoldásstruktúrában, emiatt tehetünk egy új felső korlátot ezen halmazokban levő y_i összegeire. Teljesülni kell minden ilyen esetben, hogy :

$$\forall ox \in OX : \sum_{o_j \in ox} y_i \leq 1$$

3.6. Konklúzió a kiterjesztésről

Az alábbi fejezetben láthattuk, milyen megfontolásokból valósítható meg a egy PNS probléma kiterjesztése kölcsönösen kizáró halmazokkal, és milyen változtatásokat kellett elvégezni az eredeti P-gráf keretrendszeren, hogy annak algoritmusai hatékonyan figyelembe tudják venni ezeket az új megkötéseket. Mint azt vizsgáltuk, új korlátozások bevezetése keresési tér csökkenést eredményezhet, melyet az algoritmusok bemutatott módosításával hatékonyan ki tudunk használni a megoldási idő csökkentésére, valamint megoldás során generált rész-problémák számának csökkentésére.

4. A kölcsönös kizárás által biztosított gyorsítás kihasználása feladatmegoldásban

Egyes feladatok struktúráit vizsgálva megfigyelhetjük, hogy azok tartalmaznak kölcsönösen kizáró halmazokat, azonban ezekre nem fordítunk elég figyelmet a modellalkotás, valamint a megoldási módszer választás közben. Az alábbi fejezetben azt fogom vizsgálni, és szakirodalmi példákon keresztül bemutatni, hogy ilyen esetekben a kölcsönös kizárás által nyújtott keresési tér csökkenéseket ki lehet-e használni? Továbbá ha igen, hogyan lehet hatékonyan kihasználni, és hogyan jelennek meg ezek a hatások a feladat megoldásának hatékonyságában. A vizsgálatokat járműhozzárendelési problémákon keresztül fogom reprezentálni.

4.1. Játműhozzárendelési probléma

A jármű hozzárendelési probléma (Vehicle Assignment Problem -VAP) egy specializált hozzárendelési probléma, melyben különböző elvégzendő szállítási feladatok elvégzésének céljából rendelünk hozzájuk járműveket. A problémában az elvégzendő szállítások, a rendelkezésre álló járművek, valamint a környezet (pl. távolságok, helyek ...), és minden egyéb a feladat során szükséges tulajdonság (pl. jármű kapacitás, max sebesség, fogyasztás ...) előre definiált, továbbá az is, hogy mely jármű mely feladatok elvégzésére képes. A feladatunk egy olyan hozzárendelés megvalósítása, melyben az összes szükséges feladat el lesz végezve, és valamely előre meghatározott szempontból optimális. A feladatnak többféle megoldási módja is megtalálható a szakirodalomban, a hurisztikus módszereken keresztül a bizonyítottan optimális eredményt szolgáltatóig. Röviden szeretnék bemutatni egy olyan bizonyítottan optimális megoldási módszer, mely PNS problémaként kezeli a VAP problémát, és így képes a hatékony megoldás meghatározására ([1],[2]). Ezt a módszert vizsgáljuk meg jobban, és az általa generált modellt egészítsük majd ki a kölcsönösen kizáró halmazokkal.

4.2. Játműhozzárendelési probléma mint PNS feladat

A modellkonverzió alapelgondolása a következő. Tekintsünk az elvégzendő szállítási feladatokra úgy, mint végeredmények. A nyersanyagok pedig legyenek a használható járművek a kiindulási lokációjukban. A köztes anyagok jelöljék a szállítások során érinthető lokációkat, esetleg átpakolási, vagy pihenőhelyeket, amiket érintenie kell, vagy érinthet az adott jármű minden esetben. Műveleti egységek ezen analógia alapján az egyes lokációk közti lehetséges mozgásokat fogják reprezentálni. Műveleti egységből megkülönböztethetünk három különböző fajtát. Az első fajta az egyes járműveket használatba hozza, a második fajta műveleti egység felelős a járművek lokációk közti mozgatásáról, a harmadik féle műveleti egység pedig a járművek célba juttatását végzi egy olyan lokációból, amelyből vezet út a kívánt célhoz. A modellkonverzió pontos leírása megtalálható [2]. Ezek alapján egy VAP problémát képesek leszünk PNS problémaként kezelni. A modellkonverziós eljárás pseudo kódja látható a 6.ábrán. Bementéül a VAP járműhozzárendelési feladatot várja, kimenetén pedig visszatér a generált PNS modellt.

A továbbiakban a módszer által generált PNS modellt fogjuk használni, és azt kiterjeszteni kölcsönösen kizáró halmazokkal. Azt, hogy ez miért lesz hasznos, a következő részben fogom bemutatni.

```

input:  $(T, S, A, t_s, t_e, l_s, l_e, d, l, v_{max})$  assignment problem
output:  $(\mathcal{P}, \mathcal{R}, \mathcal{O})$  process synthesis problem
begin
   $\mathcal{P} := T; \mathcal{R} := S; \mathcal{O} := \emptyset;$ 
  for all  $P_i \in T$  do
    for all  $R_k \in A(P_i)$  do
       $\mathcal{O} := \mathcal{O} \cup \{(\{R_k\}, \{R_k \cdot l_s(P_i) \cdot t_s(P_i)\}), (\{R_k \cdot l_s(P_i) \cdot t_s(P_i)\}, \{R_k \cdot l_e(P_i) \cdot t_e(P_i), P_i\})\};$ 
    end for
    for all  $P_j \in T$  do
      for all  $R_k \in A(P_i) \cap A(P_j)$  do
        if  $t_e(P_i) \leq t_s(P_j)$  and  $\frac{d(t_e(P_i), t_s(P_j))}{t_s(P_j) - t_e(P_i)} \leq v_{max}(R_k)$  then
           $\mathcal{O} := \mathcal{O} \cup \{(\{R_k \cdot l_e(P_i) \cdot t_e(P_i)\}, \{R_k \cdot l_s(P_j) \cdot t_s(P_j)\})\};$ 
        end if
      end for
    end for
  end for
end

```

6. ábra. A VAP-PNS modellkonverziót végző algoritmus pszeudo kódja

4.3. Kölcsönös kizárás értelmezése VAP feladatokban

Mint azt a korábbiakban láthattuk, a VAP problémának előállítható PNS modellje, mely a P-gráf keretrendszer segítségével, egyszerűen kezelhető, valamint a rendelkezésre álló algoritmusokkal meghatározható az optimális megoldása, és akár az n-legjobb megoldása is valamely szempontból. Azonban ez a fajta megoldási mód figyelmen kívül hagyja a kölcsönösen kizáró halmazokat, és nem veszi őket figyelembe a modell generálása során sem. A kölcsönösen kizáró halmazok modellbe foglalása nem befolyásolja a kapott eredményeket optimalitás szempontjából, csupán a megoldási időt, valamint a megoldás során generált részproblémák számát képes jelentős mértékben csökkenteni. Ez a bevezetett PNS megoldási módszernek köszönhető, mely képes figyelembe venni a kölcsönös kizárást.

A továbbiakban bemutatom, hogy hol találkozhatunk kölcsönösen kizáró művelei egység halmazokkal egy VRP-PNS modellben.

Elsőként logikai megfontolásként megemlítendő, hogy a hozzárendelések során egy időben egy jármű nem lehet egyszerre több helyen (nem hajthat végre egyszerre utazást két közbűlő lokáció között egy azon időintervallumban), ez fizikai képtelenség lenne.

Mint azt korábban említettem VRP PNS modellje 3 fajta műveleti egységet tartalmazhat. Vizsgáljuk meg ezeket.

Az első fajtába azok estek, melyek a kiinduló helyükről mozdítják el a kívánt járművet egy köztes lokációra. Ha egy jármű képes a kiinduló lokációjáról több köztes lokációba szállítani, és ezek közt találhatók olyanok, melyek szállítási időintervall metszete létezik, akkor ezekre a műveleti egységekre tekinthetünk mint kölcsönösen kizáró elemekre.

Ugyanez az analógia alkalmazható a második és a harmadik csoportba eső műveleti egységekre is. A második csoport elemei akkor kölcsönösen kizáróak, ha közös kiindulási pontból szállít ugyanaz a jármű, és a szállítások időintervallumának van metszete. Emellett még abban az esetben is, ha különböző lokációkból szállítana ugyanaz a jármű egy közös célba, valamint a legszélsőséges eset, amikor ugyanaz a jármű ugyanazon időben különböző kiindulási

pontból szállítana különböző célokba. Azon műveleti egységek halmaza, melyek valamely esetet megvalósítja, elemeinek páronként kölcsönösen kizárónak kell lennie.

A harmadik csoportba eső műveleti egységek közül, melyek a célba szállítást reprezentálják, akkor lesznek kölcsönösen kizáróak, ha különböző köztes helyről szállítanak, és a szállítási időben szintén van közös rész. Ezek mellett egyes specifikusabb problémák esetén további kölcsönösen kizáró műveleti egységeket is találhatunk csupán a struktúra, és a feladatleírás vizsgálatával. Az általam felsorolt esetek azok voltak, melyek ha megjelennek biztosan kölcsönös kizárást vonnak maguk után.

4.4. Tesztelés szakirodalomból vett példák

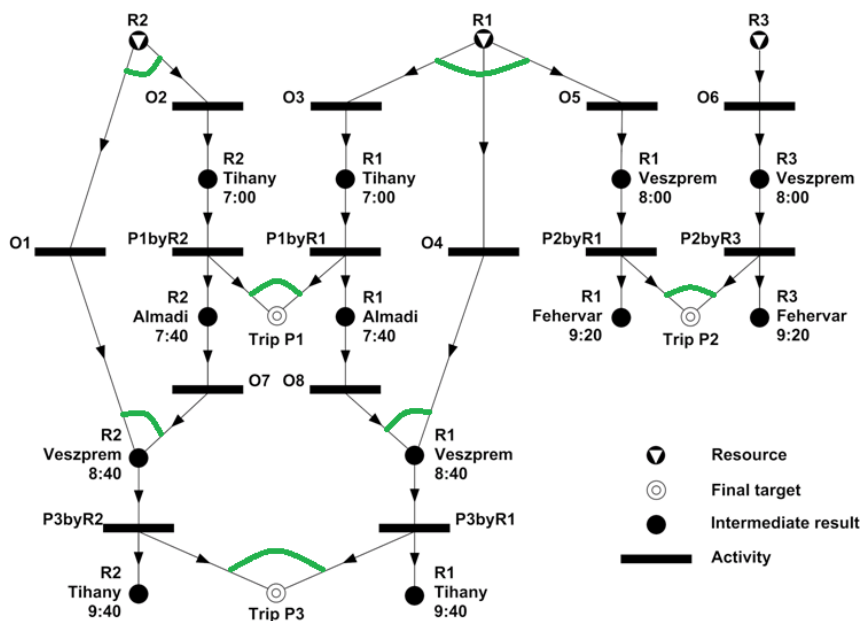
A tesztelést szakirodalomból vett példákra végeztem ([1],[2]). Fontos először megemlíteni, hogy mi is a tesztelés célja. A kölcsönös kizáró halmazok mint korábban említettem nem befolyásolják a kapott megoldások értékét, mivel ezek csak a struktúra felépítéséből és a feladat megfogalmazásából kerülnek ki. Az eddigi megoldások mindaddig olyanok voltak, amik eleget tettek a bevezetett kölcsönös kizárás kritériumoknak, csupán a különbség az, hogy a megoldási módszer nem vette őket figyelembe, és különböző korlátozások bevezetésével jött ki a jó eredmény. A PNS képessé tétele kölcsönösen kizáró halmazok jelenlétének kiszűrésére a megoldás során jelentős gyorsítást eredményezhet. Azonban megemlítendő, a megoldás számértékileg nem fog változni. A kidolgozott módszertől tehát jelentős megoldási idő gyorsulást várunk, nem pedig az optimális megoldás javítását.

4.4.1. 1. példa

Első példaként nézzünk egy olyan VRP feladat, melyben 3 elvégzendő feladat van ($P1, P2, P3$), és ehhez 3 különböző jármű áll rendelkezésre ($R1, R2, R3$) ([2]). A feladat leírása alapján a VPA-PNS modellkonverziós eljárás 14 darab műveleti egységnek megfelelő taszkot azonosított. A generált PNS feladat maximális szuperstruktúráját a 7.ábrán láthatjuk, melyen már ábrázoltam a struktúrában fellelhető kölcsönösen kizáró műveleti egység halmazokat is.

Ezek azonosítását a bevezetőben említett szempontok alapján végeztem. $\{O_1, O_2\}$, $\{O_3, O_4, O_5\}$ halmazok azért lesznek kölcsönösen kizáróak, mert páronként közös kiindulási helyről indulnak (ugyanaz a jármű) és olyan időpontokban kell megérkezniük, hogy az úton töltött időintervallumoknak lenne metszete, így az adott jármű egyszerre nem lehet több helyen. $\{P3byR2, P3byR1\}$, $\{P1byR2, P1byR1\}$ valamint $\{P2byR1, P2byR3\}$ műveleti egység halmazok azért lesznek kölcsönösen kizáróak, mert ugyanazt a feladat megvalósítását modellezzik, és egy feladatot egyszer kell csak elvégezni. Továbbá kölcsönösen kizáróak lesznek az $\{O_1, O_7\}$, $\{O_4, O_8\}$ halmazok is, mivel ezek olyan kettes csoportba tartozó műveleti egységek, melyek ugyanazt a járművet mozgatják, és célállomásuk, valamint odaérkezési idejük megegyezik, ha egyszerre szerepelének akkor az adott járműnek egyszerre több helyen kéne lennie, ami ellentmondás. Az azonosított kölcsönösen kizáró halmazok megtalálhatók a 8.ábrán.

Összesen 7 kölcsönösen kizáró műveleti egység halmazt azonosítottam a példa során, melyeket jelöltem a 7.ábrán. Valamint az alkalmazott PNS modellt kiegészítettem velük, mely modell megtalálható az 1. Függelékben. Ezt



7. ábra. Az első feladat maximális szuperstruktúrája

követően újra megoldottam a problémát. A kapott eredményeket a 1.táblázatban találhatjuk.

Megoldás #	VAP-PNS modell			+ Kölcsönös kizárás		
	P1	P2	P3	P1	P2	P3
1	R1	R3	R1	R1	R3	R1
2	R1	R3	R2	R1	R3	R2
3	R2	R1	R2	R2	R1	R2
4	R2	R3	R2	R2	R3	R2
5	R2	R3	R1	R2	R3	R1
Teljes Idő (sec.)	0.035 s			0.011 s		
Részproblémák (db.)	173			45		

1. táblázat. A számított hozzárendelések, valamint a számítás teljes ideje és a generált részproblémák száma az első tesztfeladathoz

A példára az első 5 legjobb megoldást határoztam meg mind az alap VPA-PNS modell alapján (ahogyan a szakirodalomban is végezték a számítást), valamint az általam felírt kölcsönös kizárással kiegészített modellre is a kölcsönös kizárást figyelembe vevő PNS megoldási módszer alapján. A kapott eredmények láthatók a 1.táblázatban. Az egyes sorok a számított járműhozzárendeléseket tartalmazzák külön-külön a módszerekhez (mely P_i feladatvégzését mely R_i jármű fogja végezni). Mind az 5 megoldás egyezést mutatott, azonban a futási időkben, valamint a megoldás során generált részproblémák számában jelentős eltérés tapasztalható. Az eredeti módszer, amely nem képes kölcsönös kizáró hal-

$$\begin{aligned}
ox_1 &= \{O_1, O_2\} \\
ox_2 &= \{O_3, O_4, O_5\} \\
ox_3 &= \{P3byR2, P3byR1\} \\
ox_4 &= \{P1byR2, P1byR1\} \\
ox_5 &= \{P2byR1, P2byR3\} \\
ox_6 &= \{O_1, O_7\} \\
ox_7 &= \{O_4, O_8\}
\end{aligned}$$

8. ábra. Az első feladatban azonosított kölcsönösen kizáró halmazok

mazok alapján elvetni részproblémákat 0.035 másodperc alatt oldotta meg 173 részprobléma vizsgálatával. Ezzel szemben az új, kölcsönös kizárást figyelembe vevő módszernek a kibővített modell alapján sikerült ugyanazon eredményeket 0.011 másodperc alatt, csupán 45 részprobléma vizsgálatán keresztül szolgáltatnia. A feladat eredeti megoldásait láthatjuk a 9.ábrán.

Solution	Assignments			Mileage			Total mileage	Cost	CO ₂ emission	Max running
	P1	P2	P3	R1	R2	R3	[km]	[€]	[kg]	time [min]
#1	R1	R3	R1	150	0	140	290	146	10.20	160
#2	R1	R3	R2	40	140	140	320	150	11.30	140
#3	R2	R1	R2	140	290	0	430	201	16.20	240
#4	R2	R3	R2	0	290	140	430	218	15.90	180
#5	R2	R3	R1	120	180	140	440	229	17.20	240

9. ábra. Az első feladat első 5 legjobb megoldása

4.4.2. 2. példa

A második teszt példa jóval nagyobb feladatot reprezentál. Itt 6 elvégzendő feladathoz 6 lehetséges jármű szerepel, melyhez a VAP-PNS modellgeneráló 174 műveleti egységnek megfelelő taszkot azonosított a komplett feladat, és megkötések alapján, mellyel pontosan lehet modellezni a feladatot. Ezen műveleti egység halmazok közül 36 darab kölcsönösen kizáró halmazt írtam fel a modellbe az ismertetett eseteket figyelembe vételét követően meghatározásukra. A feladat nagy méretéből következik, hogy nagyon sok megoldása van. Azonban kezelhetőség, és átvizsgálhatóság szempontjából csak az első legjobb 500-at tartottam fontosnak legenerálni mindkét módszerrel, majd ezeket hasonlítottam össze. Gyakorlati szempontból 500 alternatív megoldást nem egyszerű már összehasonlítani, ezért döntöttem úgy, hogy ennél többet nem vizsgállok. Nagyobb számú megoldás keresésénél egyébként a megoldási módszerek közti gyorsaságkülönbség csak nőtt volna. A felírt modell, amely tartalmazza a kölcsönösen kizáró halmazokat is, megtalálható a dolgozat 3.számú függelékében.

N	VAP-PNS modell		+Kölcsönös kizárás	
	Teljes idő(sec)	Részproblémák(db)	Teljes idő(sec)	Részproblémák(db)
1	0.015 s	13	0.014 s	13
5	0.112 s	119	0.054 s	95
10	0.208 s	227	0.102 s	181
50	0.982 s	1107	0.351 s	599
200	3.895 s	4435	1.360 s	2341
500	9.564 s	10983	3.048 s	5267

2. táblázat. A két különböző modellel számított N legjobb megoldás meghatározásának futási ideje, valamint a futás közben generált részproblémák száma.

A kapott eredményeket láthatjuk a 2. táblázatban. A két módszer mindegyike minden esetben ugyanazokat a megoldásokat szolgáltatta. Azonban megfigyelhető a táblázat adatai alapján, hogy egyre nagyobb N értékekre a megoldások időtartamai közti különbség jelentősen növekszik a két módszer között. Ezáltal egyre nagyobb hatékonyságnövelés érhető el a kölcsönösen kizáró halmazok figyelembevételének köszönhetően. Amint az a táblázatból kiolvasható az első 500 legjobb megoldás új modell alapján a kölcsönös kizárást figyelembe vevő algoritmussal csupán 3.048 másodpercbe telt, az eredeti módszer ugyanezt az eredményt 9.564 másodperc alatt tudta szolgáltatni. Mindezekmellett az sem elhanyagolható, hogy a megoldás során vizsgált részproblémák számát sikerült jelentősen csökkenteni. A gyorsabb megoldás nagymértékben ennek köszönhető.

4.5. Tesztelési eredmények kiértékelése

Mint azt láthattuk a bemutatott példákban, a kölcsönös kizárást figyelembe vevő megoldó algoritmus rendkívül jól képes kihasználni a korlátozások nyújtotta keresési tér csökkenést. Ez megmutatkozik a megoldás során generált részproblémák drasztikus csökkenésén, valamint a megoldási idő jelentős megrövidülésén is az eredeti megoldási módszerhez képest. Tehát leszűrhető, hogy a kölcsönösen kizáró struktúrarész-halmazok külön felismerése, és modellbe illesztése jelentősen javítja a probléma megoldásának hatékonyságát. Így amikor csak lehet próbáljunk meg ilyen halmazokat azonosítani, és modellbe foglalni, még ha alapról a feladat ezt nem is teszi meg, csak a struktúrájából következik. Fontos továbbá megjegyezni, hogy a kapott eredmények a tesztelés során minden esetben meg is egyeztek.

5. Hatványhalmaz generálás kölcsönösen kizáró részhalmazok esetén

Mint azt már korábban említettem a PNS feladat erősen kombinatorikus jellegű. Például amikor azt vizsgáljuk, hogy hogyan lehet előállítani egy m anyagot, akkor nem elég csak az említett $\varphi^+(m)$ halmaz elemeit vizsgálni. A $\varphi^+(m)$ halmazban azok a műveleti egységek szerepelnek, melyek képesek előállítani az m anyagot. Egy-egy anyagot egyszerre több műveleti egység is előállíthat, ha képes rá, és így az összes különböző kombinációt figyelembe kell venni ahhoz, hogy az össze lehetséges megoldásstruktúrához eljussunk. Ezek nem lesznek mások, mint $\varphi^+(m)$ halmaz összes lehetséges részhalmaza. Jelen esetben emiatt a $\varphi^+(m)$ halmaz hatványhalmaza fogja tartalmazni az összes lehetséges módot az üres halmazt leszámítva.

5.1. Nagyságrendi különbségek megjelenése

A továbbiakban vizsgáljuk meg, mi történik abban az esetben, ha a $\varphi^+(m)$ halmaz tartalmaz kölcsönösen kizáró részhalmazokat. Ha fennáll ez az eset, akkor nyilván azokat a hatványhalmaz elemeket, amelyek tartalmazznak legalább egy olyan kölcsönösen kizáró részhalmazt, melynek számossága nagyobb mint egy, nem tekinthetünk jó kombinációnak, mert nem teljesíti a kölcsönös kizárásra felállított feltételünket. A probléma megoldható lenne úgy is, hogy először legeneráljuk az egész hatványhalmazt, majd utána kiszűrjük belőle azokat az elemeket, melyek nem tesznek eleget a kölcsönös kizárásnak. Azonban a hatványhalmaz generálás rendkívül költséges művelet, mivel a hatványhalmaz számossága az alaphalmaz számosságával exponenciálisan változik. Egy n elemű halmaznak 2^n elemű hatványhalmaza van. Egy szélsőséges példán szemléltethető a felesleges vizsgálatok száma bizonyos esetekben. Vegyünk egy olyan 15 elemű halmazt, amely csak egymással kölcsönösen kizáró elemeket tartalmaz, és számítsuk ki ennek a halmaznak a hatványhalmazát. Mivel minden elem kölcsönösen kizáró, így csak az egy elemű halmazok fognak eleget tenni a kölcsönös kizárásnak, melyekből 15 darab van. Ezzel szemben a teljes hatványhalmaz $2^{15} = 32768$ elemet tartalmaz. Tehát a 32768 elemből összesen 15 darab olyan lesz, amely eleget tesz a feltételeinknek. Mint azt a példán is láthatjuk a 32768 elem előállítása és külön külön vizsgálata rendkívül sok műveletet felemészt ahhoz képest, hogy a jelen esetben csak 15 megoldás lehetséges. Nagyobb halmazokra a különbség még lényegesebb lehet.

5.2. Kizárási feltételnek eleget tevő részhalmazok számának becslése

A továbbiakban bemutatom, hogyan változik a lehetséges kölcsönös kizárásnak eleget tevő hatványhalmaz elemek száma, az alaphalmaz mérete, és a kölcsönösen kizáró részhalmazok számosságának tükrében.

Abban az esetben, ha a kölcsönösen kizáró halmazok között vannak olyanok, melyek tartalmazznak azonos elemeket, könnyen belátható, hogy több lehetséges megoldás van, mivel több különböző kombináció állítható elő az elemekből, így a vizsgálatom mint egy alsó becslést ad a jó hatványhalmaz elemek számára. Ennek alapján mindenre kiterjedő felső becslést is fogok majd adni. A tovább-

biakban tekintsünk úgy a kölcsönösen kizáró halmazokra, hogy azok páronként diszjunktak. Azt az esetet, melyben kölcsönösen kizáró halmazok páronként nem diszjunktak, ez a levezetés után fogom taglalni. Azonban ez az eset a gyakorlatban ritkán jelenik meg. Ha nem áll fent, akkor viszont az alsó korlátra adott becslésem a várható kölcsönös kizárást kielégítő részhalmozok pontos számát adja meg.

Példaként vegyünk egy n elemű halmazt, amely j darab kölcsönösen kizáró halmazt tartalmaz, melyek számosságai rendre $k_1 \dots k_j$. Nyilvánvalóan

$$\sum_{i=1}^j k_i \leq n,$$

vagyis a részhalmozok számosságainak összege nem haladhatja meg az alaphalmaz méretét. Továbbá feltehetjük azt is, hogy minden kölcsönösen kizáró részhalmoz számossága nagyobb mint 1, mivel ebben az esetben jelenik csak meg a kizárás.

$$\forall k_i \geq 2$$

Mint azt már említettem, egy n elemű halmaz hatványhalmaza 2^n ha nem tartalmaz kölcsönös kizáró részeket. Ez megfelel a $j = 0$ esetnek. Tehát alapesetben a hatványhalmaz számossága:

$$|\mathcal{P}(n)| = 2^n \quad (1)$$

Ha egy alaphalmaz méretét növeljük egy elemmel, az így generált hatványhalmaz mérete a duplájára növekedik, az analódia alapján, hogy az eredeti halmaz összes eddigi részhalmozába bele vesszük-e a hozzáadott elemet, vagy nem. Terjesszük ki ezt a gondolatmenetet úgy, hogy mi történik, ha egy halmaz részhalmozát kölcsönösen kizáróként definiáljuk.

Tekintsük a $j = 1$ esetet, amikor a halmaz mérete n , a definiált részhalmoz számossága pedig k , ekkor a nem kizáró rész mérete $n - k$. Ebben az esetben a hatványhalmaz elemei maradnak azok a részhalmozok is, melyek nem tartalmaznak kölcsönösen kizáró részt, ebből a jelen esetben 2^{n-k} darab lesz. Nyilvánvalóan a kölcsönös kizáró részből 1 elemnél egyszerre többet nem adhatunk hozzá egyetlen részhalmozhoz sem, mert akkor nem teljesíti a feltételeket. Az eddigi halmazokhoz most is csak egyszerre egy elemet adhatunk hozzá, azonban különbséget kell tenni, melyiket, így egy-egy részhalmozból k különböző új részhalmozot generálunk. Tehát összesen $2^{n-k} \cdot k$ darabot. A kapott hatványhalmaz mérete :

$$2^{n-k} + 2^{n-k} \cdot k = 2^{n-k} \cdot (k + 1) \quad (2)$$

Tekintsük a $j = 2$ esetet, amikor a halmaz mérete n , a definiált részhalmozok számossága pedig k_1 és k_2 , ekkor a nem kizáró rész mérete $n - k_1 - k_2$. Ebben az esetben is a hatványhalmaz elemei maradnak azok a részhalmozok is, melyek nem tartalmaznak kölcsönösen kizáró részt, ebből a jelen esetben $2^{n-k_1-k_2}$ darab lesz. Ehez jön hozzá, amikor csak egy-egy halmazt veszünk figyelembe, ez megfelel az előbb vizsgált $j = 0$ esetnek, és így $2^{n-k_1-k_2} \cdot k_1$ valamint $2^{n-k_1-k_2} \cdot k_2$ ilyen hozzáadott részhalmoz lesz. Emellett hozzáadhatunk egy-egy alap részhalmozhoz különböző halmazból vett elemeke kombinációját. Összesen $k_1 \cdot k_2$ ilyen kombináció van, mellyel újabb $2^{n-k_1-k_2} \cdot k_1 \cdot k_2$ részhalmozot adtunk hozzá a hatványhalmazhoz. A kapott hatványhalmaz mérete :

$$2^{n-k_1-k_2} + 2^{n-k_1-k_2} \cdot k_1 + 2^{n-k_1-k_2} \cdot k_2 + 2^{n-k_1-k_2} \cdot k_1 \cdot k_2 \quad (3)$$

Ami nem más mint:

$$2^{n-k_1-k_2} \cdot (k_1 + 1) \cdot (k_2 + 1) \quad (4)$$

Ezt követően írjuk fel általános alakban, a használt jelölésekkel. Az alaphalmaz n elemű, amely j darab kölcsönösen kizáró részhalmazt tartalmaz melyek számosságai $k_1 \dots k_j$. Ekkor a lehetséges kölcsönös kizárásnak megfelelő részhalmazok száma:

$$2^{n-k_1-\dots-k_j} \cdot (k_1 + 1) \cdot \dots \cdot (k_j + 1) = 2^{(n-\sum_{i=1}^j k_i)} \cdot \prod_{i=1}^j (k_i + 1) \quad (5)$$

Ezt követően vizsgáljuk meg a kiinduló $j = 0$ állapot, és az általános állapot közti eltérést, valamint azt, hogy milyen az arány a kettő között j és $k_1 \dots k_j$ -k függvényében. Osszuk le az általános állapotot a kiindulóval (vagyis $\frac{(1)}{(5)}$). Ami nem más mint:

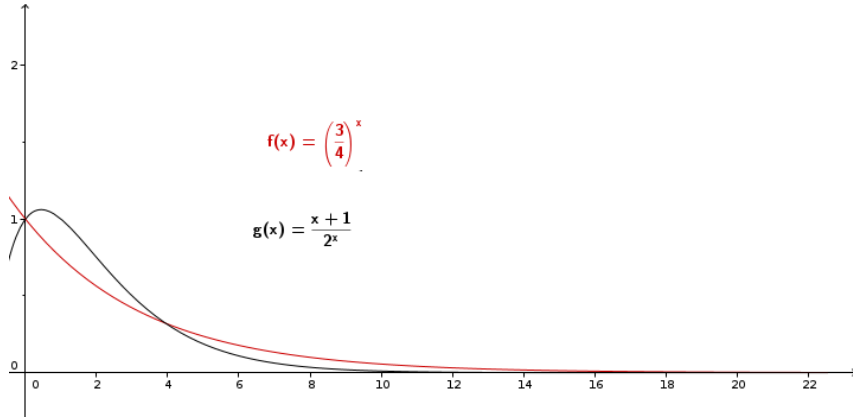
$$\frac{2^{(n-\sum_{i=1}^j k_i)} \cdot \prod_{i=1}^j (k_i + 1)}{2^n} = \frac{\prod_{i=1}^j (k_i + 1)}{2^{(\sum_{i=1}^j k_i)}} = \prod_{i=1}^j \frac{k_i + 1}{2^{k_i}} \quad (6)$$

Kikötöttük, hogy minden k_i -re igaz, hogy $k_i \geq 2$, mivel csak ebben az esetben van értelme kölcsönös kizárásról beszélni. Ilyen k_i érték mellett azonban a $\frac{k_i+1}{2^{k_i}}$ függvény szigorúan monoton csökkenő. Maximumát $k_i = 2$ helyen veszi fel, és az értéke $\frac{3}{4}$. Ezt követően térjünk ki j értékének változására. A j érték jelöli mennyi kölcsönösen kizáró halmaz lehet. Egy ilyen halmaz számossága legalább 2. Így n elemű alaphalmaz esetén maximum $\lfloor \frac{n}{2} \rfloor$. Mivel a produktum összes eleme 1-nél kisebb ($k_i = 2$ esetén határoztuk meg a maximumot). Így maga a szorzat is j növekedésével kisebb, és kisebb lesz. Maximuma abban az esetben lesz, ha az összes k_i érték a 2 értéket veszi fel. Ekkor a maximuma $(\frac{3}{4})^j$ lesz. Különböző és 2-től eltérő k_i értékekre esetén ettől még kisebb.

$$\prod_{i=1}^j \frac{k_i + 1}{2^{k_i}} \leq \left(\frac{3}{4}\right)^j$$

Konklúzióként levonható, hogy ha van kölcsönös kizárás, akkor kijelenthető, hogy az alaphalmaz hatványhalmazának maximum a $\frac{3}{4}$ -ed része lehet olyan, amely kielégíti a kölcsönös kizárást. Ez az eset $j = 1$, és $k = 2$ esetében áll fent. Minden egyéb esetben ettől jelentős mértékben kisebb a jó megoldások aránya a függvények szigorú monoton csökkenése miatt ami látható az 10. ábrán. Emellett még az is leszűrhető, hogy ha a kölcsönösen kizáró halmazok számát növelem, akkor j függvényében $(\frac{3}{4})^j$ mértékben, míg ha egy halmaz méretét bővítem ki, akkor pedig k_i függvényében $\frac{k_i+1}{2^{k_i}}$ mértékben csökken a megfelelő részhalmazok száma. Mivel mindkét függvény exponenciálisan változik, és alapjuk kisebb mint 1, azért minél nagyobb értékre értelmezzük, annál drasztikusabb csökkenést eredményez. Továbbá a j és k_i értékek együttes változtatása ezek alapján szintén exponenciális csökkenést eredményez, amely mértéke látható a (6). egyenleten. Azonban meghatároztam, hogy ez az érték nem csökkenhet akármennyire, mindig legalább

$$2^{(n-\sum_{i=1}^j k_i)} \cdot \prod_{i=1}^j (k_i + 1)$$



10. ábra. Függvények ábrázolása

darab olyan részhalmaznak kell lennie, mely kielégíti a kölcsönös kizárásra tett feltételeket.

A felállított alsó korlátunk a legtöbb esetben magát a pontos eredményt is megadja, a felállított felső korlátunk pedig bizonyos esetekben nem elég szigorú. Ennek tükrében szeretnék mutatni egy eljárást, mely alkalmazható olyan esetekben is jó részhalmazok elemszámának felső becslésére, ha a kölcsönösen kizáró halmazok nem páronként diszjunktak. A jelöléseink a régiek, jelölje n az alaphalmaz, j a kölcsönösen kizáró halmazok számát, $k_1, \dots, k_j$ pedig a halmazok számosságát. Továbbá vezessünk be új változókat, melyeket jelöljünk $t_1, \dots, t_j$ -vel, értékeiket pedig a következő módon fogjuk meghatározni. Legyen t_i azoknak a halmazelemeknek a száma az ox_i kölcsönösen kizáró halmazból, melyek nem szerepeltek előtte az $ox_1, \dots, ox_{(i-1)}$ halmazokban. Formálisan:

$$t_i = |ox_i \setminus \{\bigcup_{z=1}^{i-1} ox_z\}|$$

Ekkor a felső korlát a következő lesz:

$$2^{(n - \sum_{i=1}^j t_i)} \cdot \prod_{i=1}^j (t_i + 1).$$

Az alábbiakban röviden bizonyítom, hogy miért lesz a kapott eredmény felső korlát. Vegyük észre, hogy a bevezetett új t_i változókkal egy olyan problémát tudunk modellezni, mely nem tartalmaz olyan kölcsönösen kizáró halmazokat, melynek van közös elemük, ha a t_i értékek mellé meghatározom az új ox_i^* halmazokat is. Mindezt oly módon, hogy ox_i^* lesz az a halmaz, melynek a számosságából kaptam t_i -t.

$$ox_i^* = ox_i \setminus \{\bigcup_{z=1}^{i-1} ox_z\}$$

Akkor ez a származtatott probléma viszont biztosan több elemet tartalmaz mint az előző, mivel minden jó megoldás továbbra is kikerül belőle, ám mivel csökkenttem a kölcsönösen kizáró halmazok méretét az előtte fellépő redundancia

méretével, ezért az új modell olyan megoldásokat is elfogad jóként, melyeket az eredeti kizárt volna. Ez az analógia alapján tehát több részhalmazt generál mint az eredeti, melyek között megtalálható olyan is, amit az eredeti felírás alapján ki kellett volna zárni, emelelt tartalmazza az eredeti összes elfogadható megoldását is. Így valóban felső korlátot sikerült találnom. Ezt a felső korlátot további vizsgálatnak alávetve észrevehetjük, hogy abban az esetben, ha nem tartalmaznak a kölcsönösen kizáró halmazok közös elemeket, akkor minden t_i érték pontosan k_i -vel lesz egyenlő, és ez a felső korlát is a pontos értéket fogja visszadni a lehetséges megoldások számára abban az esetben.

5.3. Példa kölcsönös kizárást tartalmazó halmaz hatványhalmazának számosságváltozására

Tekintsünk a következő példát. Az alaphalmazt jelölje o , a kölcsönösen kizáró részhalmazait pedig OX . Az egyszerűség kedvéért a kölcsönösen kizáró részhalmazok ne tartalmazzanak azonos elemeket.

$$o = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$$

$$OX = \{\{1, 2, 3\}, \{5, 7, 9\}, \{4, 8, 11, 15\}\}$$

Ekkor $n = 16$, $j = 3$, $k_1 = 3$, $k_2 = 3$, $k_3 = 4$. Magának az alaphalmaznak $2^{16} = 65536$ eleme van, ebből azonban csak :

$$2^{16-3-3-4} \cdot (3+1)(3+1)(4+1) = 2^6 \cdot 4 \cdot 4 \cdot 5 = 5120$$

az olyan megoldás, amely eleget tesz a kölcsönös kizárásnak. Ha a $\{4, 8, 11, 15\}$ részhalmazt bővítjük $\{4, 8, 10, 11, 12, 13, 14, 15\}$ -re, az új esetben $n = 16$, $j = 3$, $k_1 = 3$, $k_2 = 3$, $k_3 = 8$ állna fent, ahol a kölcsönös kizárásnak eleget tevő részhalmazok száma csupán:

$$2^{16-3-3-7} \cdot (3+1)(3+1)(7+1) = 2^2 \cdot 4 \cdot 4 \cdot 9 = 576.$$

Az első eredmény kevesebb mint tizede, a második pedig kevesebb mint a százada az eredeti hatványhalmaznak.

5.4. Új hatványhalmaz generáló eljárás

5.4.1. Motiváció

Az előbbi levezetésekben láthatjuk, hogy a kölcsönös kizárás megjelenése hatványhalmaz generálás során minden esetben csökkenti a kitételeknek eleget tevő részhalmazok számát. Még hozzá már kis értékekre is jelentősen. A probléma azonban az, hogy a szakirodalomban nem található olyan módszer, amely képes csak a megfelelő halmazokat legenerálni. Olyan módszerek vannak, melyek az összes lehetséges megoldás közül kiszűrik a jókat, azonban ezek nem hatékonyak, ahogy láthattuk hatalmas különbségek lehetnek az alaphalmaz összes részhalmazai, valamint a jó részhalmazok száma között. Munkám motivációját az adta, hogy szerettem volna kihasználni azt a keresési tér csökkenést, melyet a kölcsönös kizárás által bevezetett korlátok nyújtanak a hatványhalmaz generálásban is, és egy olyan módszert kidolgozni, amely csak a kölcsönös kizárásnak eleget tevő részhalmazokat generálja le szisztematikusan, és az annak nem eleget tevő részhalmazokat már a keresés közben figyelmen kívül hagyja.

5.4.2. Az algoritmus bemutatása

A következő részben bemutatom a kidolgozott hatványhalmaz generálási módszert, amely csak azokat a részhalmazokat generálja le, amelyek eleget tesznek a kölcsönös kizárási feltételeknek. A kidolgozott algoritmus alapötletét a korábban említett elgondolás adta amely azt vizsgálta, mi történik akkor, ha egy n elemű halmazt bővítünk egy elemmel, és hogyan kapom meg ennek az $n+1$ elemű halmaz hatványhalmazát az n elemű halmaz hatványhalmazának ismeretében?

A új hatványhalmaz generálása úgy is történhet, hogy minden egyes hatványhalmaz elemet megtartok, valamint újra hozzáadok a hatványhalmazhoz oly módon, hogy közben bővíttem a hozzáadott elemmel. Hasonló analógiát lehet követni akkor is, ha kölcsönösen kizáró halmazzal bővítünk egy halmazt, ugyanis ebben az esetben is egyszerre csak egy elemmel bővíthetem a hatványhalmazt, mert ha többel, akkor már nem fog eleget tenni a kölcsönös kizárási feltételnek. Annyi különbség jelenik meg, hogy itt a hozzáadott halmaz minden egyes elemével ezt meg kell tenni, hogy bővíttem csak vele az eredeti hatványhalmaz elemeit, és beveszem az új részhalmazokat a teljes hatványhalmazba. Egyedüli kivételes eset az amikor 2 vagy több kölcsönösen kizáró halmaznak lehetnek közös elemei, ebben az esetben szükség van további vizsgálatokra, mert különben olyan részhalmazt kaphatnánk, amely nem tesz eleget a kölcsönös kizárási feltételeknek. Erre a vizsgálatra az algoritmus harmadik fő résznél térek ki bővebben, ahol ez is történik.

Jelöljük o -val azt a halmazt, amely hatványhalmazát szeretnénk generálni, és jelölje OX azt a halmazt, amely tartalmazza a feladatban definiált kölcsönösen kizáró halmazokat. Az algoritmus három fő részből épül fel, melyek a következők: Az első rész: Az első részben felbontja az o halmazt r és $w_1 \dots w_i$ halmazokra $\forall w_i \in WI$. A felbontás során az r halmazba kerülnek azon elemek, melyek nem szerepelnek egyetlen kölcsönösen kizáró halmazban sem.

$$r = o \cap \left\{ \bigcup_{ox \in OX} ox \right\}$$

A WI halmaz elemei pedig olyan halmazok lesznek, melyek páronként diszjunktak, és csak kölcsönösen kizáró elemeket tartalmaznak. Továbbá teljesül, hogy

$$o = r \cup \left\{ \bigcup_{w_i \in WI} w_i \right\}.$$

Ez a felbontás a következőképpen zajlik. Először az o összes elemét beletesszük r -be, majd ezt követően egy cikluson belül kiválasztjuk azt az $ox \in OX$ halmazt, melynek a legnagyobb a metszete az r halmazzal, ha van ilyen akkor hozzáadjuk WI -hez az $r \cap ox$ halmazt,

$$WI = WI \cup \{r \cap ox\}$$

majd kivesszük r -ből a $r \cap ox$ halmaz elemeit.

$$r = r \setminus \{r \cap ox\}$$

Maga a ciklus addig tart, ameddig létezik olyan $ox \in OX$, amelynek metszete az r halmazzal nem üreshalmaz. Vegyük észre, hogy a ciklus végrehajtása

után minden esetben az r halmazban csak olyan elemek szerepelnek, melyek eredetileg az o halmaznak is elemei voltak, és nem szerepelnek egyetlen kölcsönösen kizáró halmazban sem, mert akkor a ciklus során kivettük volna őket az r halmazból. Valamint minden $w_i \in WI$ halmaz páronként diszjunkt és csak kölcsönösen kizáró elemeket tartalmazó lesz, mivel folyamatosan kivettük az elemeket r halmazból, és a metszetet pedig hozzáadtuk WI -hez, így nem fordulhat az elő, hogy egy elem 2 vagy több $w_i \in WI$ halmazban szerepeljen. Csak kölcsönösen kizáró elemeket pedig azért fog tartalmazni, mert a metszetet amit képeztünk folyton csak $ox \in OX$ halmazzal képeztük. Tehát megvalósítottuk az elvárt felbontását az o halmaznak.

A második fő lépés az algoritmusban egy ps halmaz összeállítása, valamint a ps halmaz hatványhalmazának előállítása a megszokott módszerrel. Megemlíteném, hogy ps halmaz számossága függ a kölcsönösen kizárt hatványok méretétől, és számosságuktól, de minden esetben jelentősen kisebb mint az eredeti o halmaz. Így az előállítandó hatványhalmaz akár nagyságrendekkel is kisebb mint az o halmaz hatványhalmaza kölcsönös kizárási feltételek nélkül. Maga a ps halmaz tartalmazni fogja r halmaz összes elemét, valamint minden egyes $w_i \in WI$ halmazból csak egy-egy tetszőleges elemet.

$$ps = r \cup \left\{ \bigcup_{w_i \in WI} e \in w_i \right\}$$

Ezt követően generáljuk a ps hatványhalmazát, melynek jele $\mathcal{P}(ps)$.

Az algoritmus harmadik fő része egy rendkívül összetett folyamat, mely két alrészre bontható, mely alrészeket minden egyes $p \in \mathcal{P}(ps)$ halmazra végre kell hajtani akkor, ha az nem csak olyan elemeket tartalmaz, amelyek nem szerepelnek kölcsönösen kizáró halmazban. Úgy állítja elő az összes lehetséges részhalmazt, hogy a p halmazban ha van elem valamely w_i ből, akkor azt az elemet kicseréli egyenként w_i összes elemére, mint ahogy bővítjük a halmazt. Ha p több w_i halmaz elemét is tartalmazza, akkor pedig az összes lehetséges kombinációra kicseréli, amelyek eleget tesznek a kölcsönös kizárásnak. Itt említeném meg, hogy ha vannak olyan kölcsönösen kizáró halmazok, melyek tartalmazzanak 2-nél több közös elemet és azon kívül más-más elemeik is vannak, akkor a kombinációk generálásánál vigyázni kell. Vegyük észre, hogy ebben az esetben lesznek olyan kölcsönösen kizáró elemek, melyek nem egy w_i halmazba kerülnek, és így a kombinációk generálása során választhatjuk ezeket az elemeket különböző halmazból. Ennek ellenére kölcsönösen kizáróak lesznek, és nem tesznek eleget a feltételnek, így nem elfogadható eredményt szolgáltatnak. Ennek fényében mutatom be a harmadik rész működését. Az első alrész összegyűjti azokat a $w_i \in WI$ halmazokat, mely eleme megjelenik az aktuális $p \in \mathcal{P}(ps)$ halmazban. Valamint eltávolítja a kölcsönös kizáró elemeket p -ből.

A második alrész feladata pedig a lehetséges kombinációk előállítása a kiválogatott w_i halmazokból. Ezt egy rekurzív eljárás valósítja meg, melynek pszeudokódja látható 5.4.2. ábrán. Először vezessük be új jelöléseket a könnyebb átláthatóság érdekében. Jelölje *inc* az algoritmus során az adott részkombinációba eddig bevett elemek halmazát. Jelölje *exc* az adott részkombináció elemei által kizárt elemek halmazát. Valamint vezessük be egy olyan leképezést a strukturális leképezések mintájára, amely visszadja egy adott elemmel kölcsönösen kizáró elemek halmazát. Ez akkor igazán lényeges, ha egy elem nem csak egy

```

procedure PowersetGeneration( $o, OX$ )
 $r = o$ 
 $i = 1$ 
while  $s \neq \emptyset$  do
     $s = r$ 
    for all  $ox \in OX$  do
         $s = r \cap ox$  where  $|r \cap ox|$  is maximal
        if  $s \neq \emptyset$  then
             $w_i = s$ 
             $i++$ 
             $r = r \cap s$ 
        end if
    end for
end while
 $ps = r \cup \{\bigcup_{w_i \in WI} e \in w_i\}$ 
for all  $p \in \mathcal{P}(ps)$  do
     $j = 1$ 
    for all  $w_i \in WI$  do
        if  $|p \cap w_i| > 0$  then
             $s_j = w_i$ 
             $j++$ 
             $p = p \setminus s_j$ 
        end if
    end for
    Generate( $s_1 \dots s_{(j-1)}, \emptyset, \emptyset, PARTIALRES$ )
    for all  $e \in PARTIALRES$  do
         $POWSET = POWSET \cup \{e \cup p\}$ 
    end for
end for
OUTPUT:( $POWSET$ )

```

11. ábra. Az új hatványhalmaz generáló algoritmus pszeudo kódja.

```

procedure Generate( $s_k \dots s_{(j-1)}, inc, exc, PARTIALRES$ )
 $s = s \setminus exc$ 
for all  $x \in s$  do
    if  $k = (j - 1)$  then
         $PARTIALRES = PARTIALRES \cup \{inc \cup x\}$ 
    else
        Generate( $s_{(k+1)} \dots s_{(j-1)}, inc \cup x, exc \cup \tau(x), PARTIALRES$ )
    end if
end for

```

12. ábra. A lehetséges kombinációkat előállító rekurzív eljárás pszeudo kódja.

kölcsönösen kizáró halmazban szerepel. Jelölje ezt a leképezést $\tau(e)$, amely szolgáltatja azoknak az elemeknek a halmazát, amely elemeket az e elem kölcsönösen kizárja.

$$\tau(e) = \bigcup_{w_i \in WI} w_i, \text{ ahol } e \in w_i \text{ teljesül.}$$

Végigiterál minden egyes meghívást követően egy w_i halmaz azon elemein, melyek nincs kizárva, beveszi a vizsgált elemet a részkombinációba, kizárja a vizsgált elemmel kizáró halmazokat, így biztosítva azt, hogy a kombináció ne tartalmazzon kölcsönös kizárást. Majt ezt követően meghívja önmagát egy másik w_i halmazra az eddig összegyűjtött elemekkel, és az eddig kizárt elemek halmazával.

$$inc = inc \cup e$$

$$exc = exc \cup \tau(e)$$

A rekurzió addig tart, ameddig létezik olyan w_i halmaz, melynek eleme szerepel az aktuálisan vizsgált p halmazban. Ha az összes w_i halmaz bejárása megtörtént, akkor elkészült egy lehetséges kombináció, és megáll a rekurzió. Egy másik megállási feltétel is van, méghozzá abban az esetben, ha az aktuálisan vizsgált p halmaznak és a kizárt elemek exc halmazának metszete üres. Ebben az esetben a rekurzió megáll, és az eddig összegyűjtött részkombinációt eldobja, mivel nem lehet azt tovább építeni, és nem lehet jó kombináció belőle.

Ha a rekurzió előállította a lehetséges kombinációkat, csak annyi maradt hátra, hogy minden kombinációhoz hozzáadjuk az aktuális p halmaz kölcsönös kizárásban nem szereplő elemeit, majd az így kapott halmazokat beletegyjük az eddig legenerált, kölcsönös kizárásnak eleget tevő részhalmazok közé. Az algoritmus teljes pszeudo kódja látható az 5.4.2.ábrán.

Megvalósítás közben felmerült heurisztikák

Az algoritmust implementáltam C++ programozási nyelven. Az implementáció során azonban további esetleges gyorsítási lehetőségekre lettem figyelmes, melyek kisebb vizsgálatok lehetnének, de bizonyos esetekben további hatékonyságnövelés érhető el velük. Ezek további vizsgálata a későbbiekben érdekes problémákat vethet fel.

Az első ilyen az volt, az algoritmus első részében a diszjunkt halmazokra való bontás. Lehetne vizsgálatokat végezni olyan esetekben, amikor több azonos elemszámú metszettel rendelkező kizáró halmazunk van, és a halmazok tartalmazznak azonos elemeket is. Ugyanis bizonyos elemek elvételekor lehet további halmazokban szereplő elemeket is kiveszünk az eredeti halmazból, így akár kevesebb halmazzal is lefedhető az eredeti probléma, ha jól választunk az azonos számosságúak közül. Például ha két kölcsönösen kizáró halmaz metszetének számossága egyenlő a maximális metszet számosságával, akkor azt lehetne érdemesebb kiválasztani, amelyiknek nagyobb a számossága.

Kódolás közben találkoztam még egy érdekes megfontolási lehetőséggel. Melyet a rekurziónál vetettem fel. Mivel a rekurzió belül mindig növekszik a kizárt elemek száma az adott struktúrából, ezt érdemes lenne megvizsgálni, hogy milyen viszonyban van ez az esedékesen kiválasztott ox_i halmazok végigjárási sorrendjétől. Lesz-e különbség, ha nagy halmazokat járunk be először, vagy esetleg a kisebb méretűekkel kezdjük ?

Ezek csupán csak felvetett kérdések, melyek megválaszolása esetlegesen képes lehetne javítani az algoritmus hatékonyságát bizonyos esetekben.

5.5. Tesztelési eredmények

A kidolgozott algoritmust implementáltam C++ programozási nyelven. Az alábbi alfejetben az új módszer hatékonyságát fogom illusztrálni példákon keresztül. Összehasonlítási alapnak az eddig alkalmazott módszert vettem, mely alapján a hatványhalmaz generálás a jeén pillanatban történik a PNS megoldóban. Úgy működik, hogy a hatványhalmaz összes elemét legenerálja, majd ezek közül kiválogatja azokat, melyek eleget tesznek az adott kölcsönös kizárási feltételeknek. Elsőként a módszert a 5.5.alfejezetben bemutatott 2 példára teszteltem. Ezt követően pedig bemutatok egy vizsgálatot egy szakirodalomból vett példára is. Emellett a példákon a tesztelés végrehajtása előtt elvégzem a pontos értéket, vagy a felső korlát számítását, melyet a korábbiakban a 5.2.alfejezetben bemutatam a várható kölcsönös kizárásnak eleget tevő részhalmazok maximális vagy pontos számára:

$$2^{(n-\sum_{i=1}^j t_i)} \cdot \prod_{i=1}^j (t_i + 1).$$

Abban az esetben, ha nem tartalmaznak a kölcsönösen kizáró részhalmazok azonos elemeket, akkor minden esetben $t_i = k_i$ -vel így elkerülhetjük a t_i kiszámítását. Egyébként :

$$t_i = |ox_i \setminus \{\bigcup_{z=1}^{i-1} ox_z\}|.$$

5.5.1. 1. példa

Az első példának tekintsük a már említett kiinduló példát. Az o alaphalmazunk a jelen esetben $n = 16$ elemű, melyre értelmezünk $j = 3$ kölcsönösen kizáró részhalmazt, melyeket az OX halmaz tartalmaz, és számosságuk $k_1 = 3, k_2 = 3, k_3 = 4$. Ebben az esetben mivel nincsenek közös elemek a kölcsönösen kizáró halmazok között, így az alsó korlát számítás pontos eredményt szolgáltat. Helyettesítsünk be a képletbe:

$$2^{(n-\sum_{i=1}^j k_i)} \cdot \prod_{i=1}^j (k_i + 1) = 2^{16-3-3-4} \cdot (3+1)(3+1)(4+1) = 2^6 \cdot 4 \cdot 4 \cdot 5 = 5120$$

A 16 elemű halmaz hatványhalmaza $2^{16} = 65536$ lehetséges részhalmazt foglal magába. Azonban mint azt kiszámoltam, csak 5120 elem fog megfelelni a kitételeknek.

A megoldás során mindkét módszerrel ugyanazt az 5120 lehetséges részhalmazt kaptuk eredményül, melyek nem tartalmaznak kölcsönösen kizáró részhalmazt. A megoldáshoz szükséges idők közti eltérés azonban jelentős. A kapott időket a 3.táblázatban láthatjuk. Az régi módszer 0.376 másodperc alatt , míg az új módszer 0.018 másodperc alatt generálta ugyanazokat a megoldásokat, mely a régi módszer megoldási idejének kevesebb mint 5%-a. Ez kicsivel több mint hússzoros gyorsulást jelent.

$$o = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$$

$$OX = \{\{1, 2, 3\}, \{5, 7, 9\}, \{4, 8, 11, 15\}\}$$

13. ábra. Az első példa elemei

	Megoldások száma (db)	Megoldási idő (sec)
Régi módszer	5120	0.376 s
Új módszer	5120	0.018 s

3. táblázat. Az első példa futási eredményei

5.5.2. 2. példa

Második vizsgált példa szintén bemutatásra került. Bővítsük meg az első példa $\{4, 8, 11, 15\}$ kölcsönösen kizáró halmazát $\{4, 8, 10, 11, 12, 13, 14, 15\}$ -re. Az új probléma leírása látható a 14. ábrán. Ebben az esetben is $n = 16$ elemű az alaphalmaz, valamint $j = 3$ kölcsönösen kizáró részhalmazzal rendelkezünk, azonban egyik számosságát a duplájára növeltük $k_1 = 3, k_2 = 3, k_3 = 8$. Figyelembe véve azt a hatékonyságbecslést, melyet az alaphalmaz mérete, a kölcsönösen kizáró halmazok száma, valamint azok számosságára tettünk, jelen esetben az első példa után még nagyobb gyorsulást várhatunk, mivel az egyik kölcsönösen kizáró részhalmaz méretét megdupláztuk. A várható jó eredmények pontos száma (mivel megint diszjunktak a kölcsönösen kizáró halmazok) pedig:

$$2^{(n - \sum_{i=1}^j k_i)} \cdot \prod_{i=1}^j (k_i + 1) = 2^{16-3-3-8} \cdot (3+1)(3+1)(8+1) = 2^2 \cdot 4 \cdot 4 \cdot 9 = 576$$

Ebben az esetben, mint azt kiszámoltam 576 olyan megoldást várunk, mely kielégíti a kölcsönös kizárást a 65536 közül.

$$o = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$$

$$OX = \{\{1, 2, 3\}, \{5, 7, 9\}, \{4, 8, 10, 11, 12, 13, 14, 15\}\}$$

14. ábra. A második példa elemei

A megoldás során mindkét módszerrel 576 olyan részhalmazt kaptunk, mely eleget tesz a kitételeknek, amely megegyezik a várt számmal. A futásidőket a 4. táblázatban láthatjuk. A régi módszer 0.377 másodperc, míg az új módszer 0.002 másodperc alatt szolgáltatja a jó eredményt. A régi módszerhez képest az új közel 200-szor gyorsabban oldotta meg ugyanazt a feladatot. Továbbá látható, hogy az első feladat megoldásánál 9-szer gyorsabb volt maga az új módszer is. A két feladat pedig csak egyetlen kölcsönösen kizáró halmazban tért el. Ezek a számok igazolták elvárásainkat. Ugyanis láthatjuk, hogy növelve

az egyik részhalmaz méretét a megoldási idő jelentősen csökken a keresési tér csökkenés miatt.

	Megoldások száma (db)	Megoldási idő (sec)
Régi módszer	576	0.377 s
Új módszer	576	0.002 s

4. táblázat. A második példa futási eredményei

5.5.3. 3. példa

Harmadik példaként tekintsünk egy kicsit nagyobb, és összetettebb problémát. Legyen az alaphalmazunk $n = 22$ elemű, amire definiálunk $j = 5$ darab kölcsönösen kizáró elemhalmazt melyeknek lesznek közös részhalmazai, és számosságuk egyenként $k_1 = 4, k_2 = 5, k_3 = 6, k_4 = 4, k_5 = 2$. Például a 2, 3, 4 elemek egyszerre megtalálhatóak mind az ox_1 , mind pedig az ox_2 kölcsönösen kizáró halmazokban is. A probléma leírását láthatjuk a 15. ábrán. Ebben az esetben, mivel vannak olyan részhalmazok, melyek kölcsönösen kizárók, így a jó megoldások számát felülről tudjuk csak becsülni. Először számítsuk ki a t_i értékeket. Melyek $t_1 = 4, t_2 = 2, t_3 = 5, t_4 = 1, t_5 = 2$ lesznek, majd helyettesítsük be őket a képletbe:

$$2^{(n - \sum_{i=1}^j t_i)} \cdot \prod_{i=1}^j (t_i + 1) = 2^8 \cdot 5 \cdot 3 \cdot 6 \cdot 2 \cdot 3 = 138240$$

A teljes hatványhalmaz mérete $2^{22} = 4194304$ ami közül tehát kevesebb mint 138240 jó megoldást várunk.

$$o = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21\}$$

$$ox_1 = \{1, 2, 3, 4\}$$

$$ox_2 = \{2, 3, 4, 6, 7\}$$

$$ox_3 = \{4, 8, 10, 11, 17, 19\}$$

$$ox_4 = \{6, 7, 18, 19\}$$

$$ox_5 = \{20, 21\}$$

$$OX = \{ox_1, ox_2, ox_3, ox_4, ox_5\}$$

15. ábra. A harmadik példa elemei

A megoldás során mindkét módszerrel 50688 olyan részhalmazt kaptunk, mely eleget tesz a kitételeknek, és jócskán ugyan, de a megadott felső korlát alatt van. A futásidőket 5. táblázatban láthatjuk. A régi módszer 33.192 másodperc, míg az új módszer 0.405 másodperc alatt szolgáltatja a jó eredményt. A régi módszerhez képest az új közel 83-szor gyorsabban oldotta meg ugyanazt

a feladatot. Mivel a két megoldási módszer megoldásai megegyeznek, így biztosan az új megoldás is megtalálta az összes kizárási feltételeknek eleget tevő részhalmazt, mert a régiben ez garantált amiatt, hogy megvizsgálja az összes lehetségeset, és azok közül szűri ki a kizárási feltételeknek eleget tevőket.

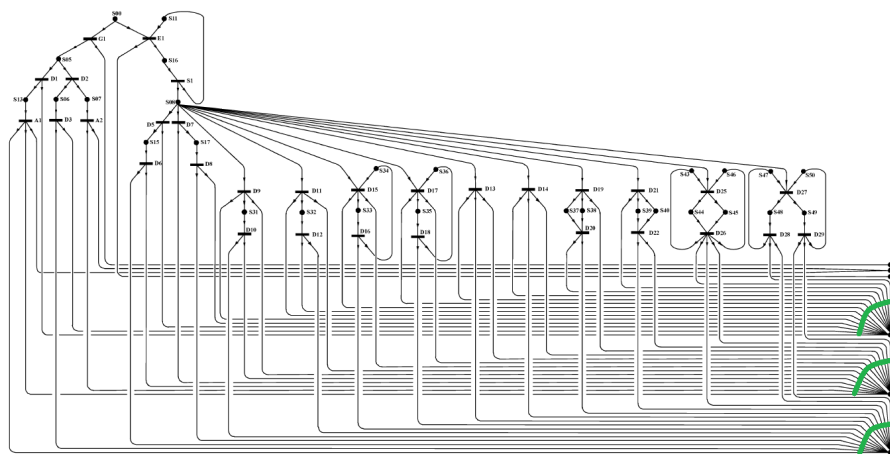
	Megoldások száma (db)	Megoldási idő (sec)
Régi módszer	50688	0.405 s
Új módszer	50688	33.192 s

5. táblázat. A harmadik példa futási eredményei

Mint azt láthattuk az első és a második példában a régi megoldási módszer sebessége lényegileg megegyezik, az új módszer pedig 9 szerez gyorsulást mutat. Ez amiatt van, hogy a régi módszer legenerálja az alaphalmaz összes részhamazát, és ezt követően azok közül szűri ki a feltételeknek nem megfelelőket. Mivel mindkét megoldás alaphalmaza egyezett, és a szűrés a hatványhalmaz generáláshoz képes nem annyira jelentős folyamat, így a megoldási idők is megegyeztek. Az új módszernél az eltérés viszont annak tudható be, mivel az csak a kölcsönös kizárásnak eleget tevő részhalmazokat generálja, így futás ideje a lehetséges jó megoldások számával arányos, nem pedig az alaphalmaz méretével. Mivel az alap hatványhalmaz generálás bonyolultsága, és megoldási ideje exponenciálisan növekszik az alaphalmaz méretével növekedésével, így egyre nagyobb és nagyobb halmazokkal történő számolás egyre több és több időt vesz igénybe. Sajnos ez már elég kis n értékekre is már jelentős, sőt akár kivárthatatlan nagyságokat ölthet.

5.5.4. 4. példa

A negyedik példában vizsgáljunk egy szakirodalomban elérhető példát [7]. Maga a példa nem nagy, ám strukturális felépítése miatt egy SSG algoritmust például nem lehet rajta lefuttatni, hogy az legenerálja az összes lehetséges kombinatorikusan lehetséges megoldásstruktúrát, mivel annyira sok részprobléma keletkezne a régi módszerekkel, hogy vagy a számítógép fizikai memóriája lenne kevés az eltárolásra, vagy pedig nem végezne vele belátható időn belül. A feladat szuperstruktúráját a 5.5.4. ábrán láthatjuk. A probléma abból adódik, hogy az SSG úgy halad előre, hogy döntéseket hogy bizonyos anyaghalmazok előállításáról. Ebben az esetben végig kell vizsgálnia az adott anyaghalmazt előállítani képes műveleti egységek halmazának összes lehetséges részhalmazát, mely a hatványhalmaza. A jelen példában ez a vizsgálat a termékektől indul, és itt kapásból 35 darab műveleti egység van, amely szóba jöhet a végtermékek legyártásába. A 35 műveleti egység azt jelenti, hogy összesen 2^{35} darab esetet kell vizsgálni, csak a kiinduló állapotban, amely 34359738368. Továbbá a szakirodalomból tudjuk, hogy összesen 3465 kombinatorikusan lehetséges megoldásstruktúra létezik. Azonban a feladat ezek között definiál kölcsönösen kizárókat, melyet ha képesek leszünk figyelembe venni, a probléma megoldható lenne jelentős időmegtakarítással. Ezen a példán keresztül láthatjuk, hogy nagy szükség lenne a P-gráf keretrendszer algoritmusában a hatványhalmaz generálást is lecserélni az általam kidolgozotttra további hatékonyságnövelés érdekében.



16. ábra. Az 4. feladat maximális szuperstruktúrája

6. Továbblépési lehetőségek

Mint azt a dolgozatból láthattuk a kölcsönös kizárást nem szabad elhanyagolni. Ha megjelenik egyes feladat struktúrákban, akkor nagy mértéken gyorsíthatja a megoldó módszer futási idejét. Azonban a PNS megoldóban további gyorsítások érhetőek el ezen a téren, ugyanis az nem az általam kidolgozott módszert tartalmazza hatványhalmaz generálás céljára kölcsönös kizárás esetén, hanem az egyszerű szűrési feltételekkel rendelkezőt, mely láthattuk kevésbé hatékony. Kutatásom további célja tehát a kidolgozott hatványhalmaz generálás beleépítése a PNS megoldó szoftverbe, ezzel megpróbálni tovább javítva a feladatmegoldás hatásfokát, és a megoldható feladatok méretének növelését.

7. Összefoglalás

Munkám során részt vettem a P-gráf keretrendszer olyan kiterjesztésének kifejlesztésében, mely képes kezelni a kölcsönösen kizáró halmazokat. Az ehez szükséges ábrázolási és algoritmus módosítási változásokat a dolgozatomban bemutattam, majd az új módszer hatékonyságát vizsgáltam járműhozzárendelési problémákon. Definiáltam olyan eseteket, melyek segítségével egy járműhozzárendelési probléma struktúrájában felismerhetünk kölcsönösen kizáró halmazokat, és ezek PNS modellbe iktatásával egy jóval hatékonyabb módszert adtam ezen feladatok megoldására, mint az eddig ismert. Ezt reprezentálják a bemutatott teszteredmények is. Kutatásom fő témája azonban a hatványhalmaz generálása kölcsönösen kizáró részhalmazok esetén volt. Alsó és felső korlátot adtam a lehetséges kölcsönös kizárásnak eleget tevő részhalmazok számosságára adott problémához, melyet bizonyítottam is. Az alsó és felső korlát mellett egy konkrét becslést is adtam a számukra, mely az esetek többségében, amikor a kölcsönösen kizáró halmazok páronként diszjunktak, pontosan megadja a várható jó megoldások számát. Emellett kidolgoztam egy olyan hatványhalmaz generáló eljárást, mely képes figyelembevenni az alaphalmazon értelmezett kölcsönösen kizáró részhalmazokat. Ezt ráadásul szisztematikus keresés segítségével végzi, mely során a kölcsönös kizárásnak nem megfelelő részproblémákat elveti, és eredményül csak olyan részhalmazokat generál, melyek eleget tesznek a kölcsönös kizárásnak. A kidolgozott algoritmust megvalósítottam C++ programozási nyelven. Valamint több teszt példán keresztül bemutattam hatékonyságát.

Hivatkozások

- [1] Barany, M., B. Bertok, Z. Kovacs, F. Friedler, and L. T. Fan, Solving vehicle assignment problems by process-network synthesis to minimize cost and environmental impact of transportation, *Clean Technologies and Environmental Policy*, 13(4), 637-642 (2011).
- [2] Barany, M., B. Bertok, Z. Kovacs, F. Friedler, and L. T. Fan, Optimization Software for Solving Vehicle Assignment Problems to Minimize Cost and Environmental Impact of Transportation, *Chemical Engineering Transactions*, 21, 499-504 (2010).
- [3] Friedler, F., K. Tarjan, Y. W. Huang, and L. T. Fan, Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems, *Chem. Engng Sci.*, 47, 1973-1988 (1992).
- [4] 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, 929-942 (1993).
- [5] Friedler, F., J. B. Varga, and L. T. Fan, Decision-Mapping for Design and Synthesis of Chemical Processes: Application to Reactor-Network Synthesis, *AIChE Symposium Series* (Eds: L. T. Biegler and M. F. Doherty), 91, 246-250 (1995).
- [6] 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, 15-20 (1995).
- [7] Liu, J., L. T. Fan, P. Seib, F. Friedler, and B. Bertok, Downstream Process Synthesis for Biochemical Production of Butanol, Ethanol, and Acetone from Grains: Generation of Optimal and Near-Optimal Flowsheets with Conventional Operating Units, *Biotechnol. Prog.*, 20, 1518 - 1527 (2004).
- [8] Friedler, F., L. T. Fan, and B. Imreh, Process Network Synthesis: Problem Definition, *Networks*, 28, 119-124 (1998).
- [9] 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), pp. 609-626, Kluwer Academic Publishers, Dordrecht, (1996).
- [10] Documents Associated with Business Process Model and Notation (BPMN) Version 2.0, www.omg.org/spec/BPMN/2.0/ <http://www.omg.org/spec/BPMN/2.0/>, Release date: January 2011

8. Függelék

Tesztelés során használt modellek

8.1. Az 1.feladat alap PNS modellje

```
file_type=PNS_problem_v1
file_name=Example

measurement_units:# comment at the end of line
mass_unit=t
time_unit=yr
money_unit=EUR#material_flow_measurement_unit=mass/time
    material_price_measurement_unit=money/mass
    operating_cost_measurement_unit=money/time

defaults:
material_type=intermediate
material_flow_rate_lower_bound=0
material_flow_rate_upper_bound=100000
material_price=0
operating_unit_capacity_lower_bound=0
operating_unit_capacity_upper_bound=100
operating_unit_fix_cost=0
operating_unit_proportional_cost=0

materials:
R1: raw_material
R2: raw_material
R3: raw_material
P1: product , flow_rate_lower_bound=1
P2: product , flow_rate_lower_bound=1
P3: product , flow_rate_lower_bound=1

operating_units:

material_to_operating_unit_flow_rates:
O1: R2 => R2_Veszprem_0840
P3byR2: R2_Veszprem_0840 => R2_Tihany_0940 , P3
O2: R2 => R2_Tihany_0700
P1byR2: R2_Tihany_0700 => R2_Almadi_0740 , P1
O7: R2_Almadi_0740 => R2_Veszprem_0840
O3: R1 => R1_Tihany_0700
P1byR1: R1_Tihany_0700 => R1_Almadi_0740 , P1
O8: R1_Almadi_0740 => R1_Veszprem_0840
P3byR1: R1_Veszprem_0840 => R1_Tihany_0940 , P3
O4: R1 => R1_Veszprem_0840
O5: R1 => R1_Veszprem_0800
P2byR1: R1_Veszprem_0800 => R1_Fehervar_0920 , P2
```

```
O6: R3 => R3_Veszprem_0800
P2byR3: R3_Veszprem_0800 => R3_Fehervar_0920, P2
```

sources/bus.in

8.2. Az 1.feladat kiterjesztett PNS modellje

```
file_type=PNS_problem_v1
file_name=Example

measurement_units:# comment at the end of line
mass_unit=t
time_unit=yr
money_unit=EUR#material_flow_measurement_unit=mass/time
    material_price_measurement_unit=money/mass
    operating_cost_measurement_unit=money/time

defaults:
material_type=intermediate
material_flow_rate_lower_bound=0
material_flow_rate_upper_bound=100000
material_price=0
operating_unit_capacity_lower_bound=0
operating_unit_capacity_upper_bound=100
operating_unit_fix_cost=0
operating_unit_proportional_cost=0

materials:
R1: raw_material
R2: raw_material
R3: raw_material
P1: product, flow_rate_lower_bound=1
P2: product, flow_rate_lower_bound=1
P3: product, flow_rate_lower_bound=1

operating_units:

material_to_operating_unit_flow_rates:
O1: R2 => R2_Veszprem_0840
P3byR2: R2_Veszprem_0840 => R2_Tihany_0940, P3
O2: R2 => R2_Tihany_0700
P1byR2: R2_Tihany_0700 => R2_Almadi_0740, P1
O7: R2_Almadi_0740 => R2_Veszprem_0840
O3: R1 => R1_Tihany_0700
P1byR1: R1_Tihany_0700 => R1_Almadi_0740, P1
O8: R1_Almadi_0740 => R1_Veszprem_0840
P3byR1: R1_Veszprem_0840 => R1_Tihany_0940, P3
O4: R1 => R1_Veszprem_0840
O5: R1 => R1_Veszprem_0800
```

```

P2byR1: R1_Veszprem_0800 => R1_Fehervar_0920, P2
O6: R3 => R3_Veszprem_0800
P2byR3: R3_Veszprem_0800 => R3_Fehervar_0920, P2

mutually_exclusive_sets_of_operating_units:
x1: O1, O2
x2: O3, O4, O5
x3: P1byR1, P1byR2
x4: P3byR1, P3byR2
x5: P2byR1, P2byR3
x6: O1, O7
x7: O4, O8

```

sources/bus_mx.in

8.3. Az 2.feladat kiterjesztett PNS modellje

```

file_type=PNS_problem_v1
file_name=Truck_Scehduling

measurement_units:# comment at the end of line
mass_unit=db
time_unit=day
money_unit=EUR#material_flow_measurement_unit=mass/time
    material_price_measurement_unit=money/mass
    operating_cost_measurement_unit=money/time

defaults:
material_type=intermediate
material_flow_rate_lower_bound=0
material_flow_rate_upper_bound=10000000
material_price=0
operating_unit_capacity_lower_bound=0
operating_unit_capacity_upper_bound=1
operating_unit_fix_cost=0
operating_unit_proportional_cost=0

materials:
CO2: raw_material
EURO: raw_material, price=1
P1: product, flow_rate_lower_bound=1,
    flow_rate_upper_bound=1
P2: product, flow_rate_lower_bound=1,
    flow_rate_upper_bound=1
P3: product, flow_rate_lower_bound=1,
    flow_rate_upper_bound=1
P4: product, flow_rate_lower_bound=1,
    flow_rate_upper_bound=1

```

```

P5: product , flow_rate_lower_bound=1,
    flow_rate_upper_bound=1
P6: product , flow_rate_lower_bound=1,
    flow_rate_upper_bound=1

operating_units:
km_R1_: capacity_upper_bound=800
km_R2_: capacity_upper_bound=800
km_R3_: capacity_upper_bound=800
km_R4_: capacity_upper_bound=800
km_R5_: capacity_upper_bound=800
km_R6_: capacity_upper_bound=800
R1_garage: capacity_upper_bound=1
P1_R1_b0800_f0915: capacity_lower_bound=1
P2_R1_t1030_f1125: capacity_lower_bound=1
P3_R1_k1100_d1340: capacity_lower_bound=1
P4_R1_s1430_b1535: capacity_lower_bound=1
P5_R1_t1350_k1625: capacity_lower_bound=1
P6_R1_v1000_e1140: capacity_lower_bound=1
R2_garage: capacity_upper_bound=1
P1_R2_b0800_f0915: capacity_lower_bound=1
P2_R2_t1030_f1125: capacity_lower_bound=1
P3_R2_k1100_d1340: capacity_lower_bound=1
P4_R2_s1430_b1535: capacity_lower_bound=1
P5_R2_t1350_k1625: capacity_lower_bound=1
P6_R2_v1000_e1140: capacity_lower_bound=1
R3_garage: capacity_upper_bound=1
P1_R3_b0800_f0915: capacity_lower_bound=1
P2_R3_t1030_f1125: capacity_lower_bound=1
P3_R3_k1100_d1340: capacity_lower_bound=1
P4_R3_s1430_b1535: capacity_lower_bound=1
P5_R3_t1350_k1625: capacity_lower_bound=1
P6_R3_v1000_e1140: capacity_lower_bound=1
R4_garage: capacity_upper_bound=1
P1_R4_b0800_f0915: capacity_lower_bound=1
P2_R4_t1030_f1125: capacity_lower_bound=1
P3_R4_k1100_d1340: capacity_lower_bound=1
P4_R4_s1430_b1535: capacity_lower_bound=1
P5_R4_t1350_k1625: capacity_lower_bound=1
P6_R4_v1000_e1140: capacity_lower_bound=1
R5_garage: capacity_upper_bound=1
P1_R5_b0800_f0915: capacity_lower_bound=1
P2_R5_t1030_f1125: capacity_lower_bound=1
P3_R5_k1100_d1340: capacity_lower_bound=1
P4_R5_s1430_b1535: capacity_lower_bound=1
P5_R5_t1350_k1625: capacity_lower_bound=1
P6_R5_v1000_e1140: capacity_lower_bound=1
R6_garage: capacity_upper_bound=1
P1_R6_b0800_f0915: capacity_lower_bound=1
P2_R6_t1030_f1125: capacity_lower_bound=1

```

```

P3_R6_k1100_d1340: capacity_lower_bound=1
P4_R6_s1430_b1535: capacity_lower_bound=1
P5_R6_t1350_k1625: capacity_lower_bound=1
P6_R6_v1000_e1140: capacity_lower_bound=1

material_to_operating_unit_flow_rates:
km_R1_: 0.8 EURO, 300 CO2 => km_R1
km_R2_: 0.8 EURO, 300 CO2 => km_R2
km_R3_: 1.2 EURO, 375 CO2 => km_R3
km_R4_: 1.2 EURO, 375 CO2 => km_R4
km_R5_: 1.5 EURO, 360 CO2 => km_R5
km_R6_: 1.5 EURO, 360 CO2 => km_R6
P1_R1_b0800_f0915: R1_b0800, 74 km_R1 => P1, R1_f0915
P2_R1_t1030_f1125: R1_t1030, 54 km_R1 => P2, R1_f1125
P3_R1_k1100_d1340: R1_k1100, 162 km_R1 => P3, R1_d1340
P4_R1_s1430_b1535: R1_s1430, 68 km_R1 => P4, R1_b1535
P5_R1_t1350_k1625: R1_t1350, 155 km_R1 => P5, R1_k1625
P6_R1_v1000_e1140: R1_v1000, 101 km_R1 => P6, R1_e1140
R1_f0915_t1030: R1_f0915, 54 km_R1 => R1_t1030
R1_f0915_s1430: R1_f0915, 17 km_R1 => R1_s1430
R1_f0915_t1350: R1_f0915, 54 km_R1 => R1_t1350
R1_f0915_v1000: R1_f0915, 44 km_R1 => R1_v1000
R1_f1125_s1430: R1_f1125, 17 km_R1 => R1_s1430
R1_f1125_t1350: R1_f1125, 54 km_R1 => R1_t1350
R1_d1340_s1430: R1_d1340, 40 km_R1 => R1_s1430
R1_e1140_s1430: R1_e1140, 53 km_R1 => R1_s1430
R1_e1140_t1350: R1_e1140, 53 km_R1 => R1_t1350
R1_garage: R1_gvi => R1_gvo
R1_gvo_b0800: R1_gvo, 120 km_R1 => R1_b0800
R1_gvo_t1030: R1_gvo, 93 km_R1 => R1_t1030
R1_gvo_k1100: R1_gvo, 77 km_R1 => R1_k1100
R1_gvo_s1430: R1_gvo, 61 km_R1 => R1_s1430
R1_gvo_t1350: R1_gvo, 61 km_R1 => R1_t1350
R1_gvo_v1000: R1_gvo, 0 km_R1 => R1_v1000
R1_f0915_gvi: R1_f0915, 44 km_R1 => R1_gvi
R1_f1125_gvi: R1_f1125, 44 km_R1 => R1_gvi
R1_d1340_gvi: R1_d1340, 93 km_R1 => R1_gvi
R1_b1535_gvi: R1_b1535, 120 km_R1 => R1_gvi
R1_k1625_gvi: R1_k1625, 77 km_R1 => R1_gvi
R1_e1140_gvi: R1_e1140, 101 km_R1 => R1_gvi
P1_R2_b0800_f0915: R2_b0800, 74 km_R2 => P1, R2_f0915
P2_R2_t1030_f1125: R2_t1030, 54 km_R2 => P2, R2_f1125
P3_R2_k1100_d1340: R2_k1100, 162 km_R2 => P3, R2_d1340
P4_R2_s1430_b1535: R2_s1430, 68 km_R2 => P4, R2_b1535
P5_R2_t1350_k1625: R2_t1350, 155 km_R2 => P5, R2_k1625
P6_R2_v1000_e1140: R2_v1000, 101 km_R2 => P6, R2_e1140
R2_f0915_t1030: R2_f0915, 54 km_R2 => R2_t1030
R2_f0915_s1430: R2_f0915, 17 km_R2 => R2_s1430
R2_f0915_t1350: R2_f0915, 54 km_R2 => R2_t1350
R2_f0915_v1000: R2_f0915, 44 km_R2 => R2_v1000

```

R2_f1125_s1430: R2_f1125, 17 km_R2 $\Rightarrow$ R2_s1430
R2_f1125_t1350: R2_f1125, 54 km_R2 $\Rightarrow$ R2_t1350
R2_d1340_s1430: R2_d1340, 40 km_R2 $\Rightarrow$ R2_s1430
R2_e1140_s1430: R2_e1140, 53 km_R2 $\Rightarrow$ R2_s1430
R2_e1140_t1350: R2_e1140, 53 km_R2 $\Rightarrow$ R2_t1350
R2_garage: R2_gdi $\Rightarrow$ R2_gdo
R2_gdo_b0800: R2_gdo, 75 km_R2 $\Rightarrow$ R2_b0800
R2_gdo_t1030: R2_gdo, 115 km_R2 $\Rightarrow$ R2_t1030
R2_gdo_k1100: R2_gdo, 162 km_R2 $\Rightarrow$ R2_k1100
R2_gdo_s1430: R2_gdo, 40 km_R2 $\Rightarrow$ R2_s1430
R2_gdo_t1350: R2_gdo, 115 km_R2 $\Rightarrow$ R2_t1350
R2_gdo_v1000: R2_gdo, 93 km_R2 $\Rightarrow$ R2_v1000
R2_f0915_gdi: R2_f0915, 57 km_R2 $\Rightarrow$ R2_gdi
R2_f1125_gdi: R2_f1125, 57 km_R2 $\Rightarrow$ R2_gdi
R2_d1340_gdi: R2_d1340, 0 km_R2 $\Rightarrow$ R2_gdi
R2_b1535_gdi: R2_b1535, 75 km_R2 $\Rightarrow$ R2_gdi
R2_k1625_gdi: R2_k1625, 162 km_R2 $\Rightarrow$ R2_gdi
R2_e1140_gdi: R2_e1140, 53 km_R2 $\Rightarrow$ R2_gdi
P1_R3_b0800_f0915: R3_b0800, 74 km_R3 $\Rightarrow$ P1, R3_f0915
P2_R3_t1030_f1125: R3_t1030, 54 km_R3 $\Rightarrow$ P2, R3_f1125
P3_R3_k1100_d1340: R3_k1100, 162 km_R3 $\Rightarrow$ P3, R3_d1340
P4_R3_s1430_b1535: R3_s1430, 68 km_R3 $\Rightarrow$ P4, R3_b1535
P5_R3_t1350_k1625: R3_t1350, 155 km_R3 $\Rightarrow$ P5, R3_k1625
P6_R3_v1000_e1140: R3_v1000, 101 km_R3 $\Rightarrow$ P6, R3_e1140
R3_f0915_t1030: R3_f0915, 54 km_R3 $\Rightarrow$ R3_t1030
R3_f0915_s1430: R3_f0915, 17 km_R3 $\Rightarrow$ R3_s1430
R3_f0915_t1350: R3_f0915, 54 km_R3 $\Rightarrow$ R3_t1350
R3_f0915_v1000: R3_f0915, 44 km_R3 $\Rightarrow$ R3_v1000
R3_f1125_s1430: R3_f1125, 17 km_R3 $\Rightarrow$ R3_s1430
R3_f1125_t1350: R3_f1125, 54 km_R3 $\Rightarrow$ R3_t1350
R3_d1340_s1430: R3_d1340, 40 km_R3 $\Rightarrow$ R3_s1430
R3_e1140_s1430: R3_e1140, 53 km_R3 $\Rightarrow$ R3_s1430
R3_e1140_t1350: R3_e1140, 53 km_R3 $\Rightarrow$ R3_t1350
R3_garage: R3_gti $\Rightarrow$ R3_gto
R3_gto_b0800: R3_gto, 54 km_R3 $\Rightarrow$ R3_b0800
R3_gto_t1030: R3_gto, 0 km_R3 $\Rightarrow$ R3_t1030
R3_gto_k1100: R3_gto, 155 km_R3 $\Rightarrow$ R3_k1100
R3_gto_s1430: R3_gto, 67 km_R3 $\Rightarrow$ R3_s1430
R3_gto_t1350: R3_gto, 0 km_R3 $\Rightarrow$ R3_t1350
R3_gto_v1000: R3_gto, 93 km_R3 $\Rightarrow$ R3_v1000
R3_f0915_gti: R3_f0915, 54 km_R3 $\Rightarrow$ R3_gti
R3_f1125_gti: R3_f1125, 54 km_R3 $\Rightarrow$ R3_gti
R3_d1340_gti: R3_d1340, 115 km_R3 $\Rightarrow$ R3_gti
R3_b1535_gti: R3_b1535, 63 km_R3 $\Rightarrow$ R3_gti
R3_k1625_gti: R3_k1625, 155 km_R3 $\Rightarrow$ R3_gti
R3_e1140_gti: R3_e1140, 53 km_R3 $\Rightarrow$ R3_gti
P1_R4_b0800_f0915: R4_b0800, 74 km_R4 $\Rightarrow$ P1, R4_f0915
P2_R4_t1030_f1125: R4_t1030, 54 km_R4 $\Rightarrow$ P2, R4_f1125
P3_R4_k1100_d1340: R4_k1100, 162 km_R4 $\Rightarrow$ P3, R4_d1340
P4_R4_s1430_b1535: R4_s1430, 68 km_R4 $\Rightarrow$ P4, R4_b1535

P5_R4_t1350_k1625: R4_t1350, 155 km_R4 $\Rightarrow$ P5, R4_k1625
 P6_R4_v1000_e1140: R4_v1000, 101 km_R4 $\Rightarrow$ P6, R4_e1140
 R4_f0915_t1030: R4_f0915, 54 km_R4 $\Rightarrow$ R4_t1030
 R4_f0915_s1430: R4_f0915, 17 km_R4 $\Rightarrow$ R4_s1430
 R4_f0915_t1350: R4_f0915, 54 km_R4 $\Rightarrow$ R4_t1350
 R4_f0915_v1000: R4_f0915, 44 km_R4 $\Rightarrow$ R4_v1000
 R4_f1125_s1430: R4_f1125, 17 km_R4 $\Rightarrow$ R4_s1430
 R4_f1125_t1350: R4_f1125, 54 km_R4 $\Rightarrow$ R4_t1350
 R4_d1340_s1430: R4_d1340, 40 km_R4 $\Rightarrow$ R4_s1430
 R4_e1140_s1430: R4_e1140, 53 km_R4 $\Rightarrow$ R4_s1430
 R4_e1140_t1350: R4_e1140, 53 km_R4 $\Rightarrow$ R4_t1350
 R4_garage: R4_gbi $\Rightarrow$ R4_gbo
 R4_gbo_b0800: R4_gbo, 0 km_R4 $\Rightarrow$ R4_b0800
 R4_gbo_t1030: R4_gbo, 54 km_R4 $\Rightarrow$ R4_t1030
 R4_gbo_k1100: R4_gbo, 187 km_R4 $\Rightarrow$ R4_k1100
 R4_gbo_s1430: R4_gbo, 68 km_R4 $\Rightarrow$ R4_s1430
 R4_gbo_t1350: R4_gbo, 54 km_R4 $\Rightarrow$ R4_t1350
 R4_gbo_v1000: R4_gbo, 120 km_R4 $\Rightarrow$ R4_v1000
 R4_f0915_gbi: R4_f0915, 74 km_R4 $\Rightarrow$ R4_gbi
 R4_f1125_gbi: R4_f1125, 74 km_R4 $\Rightarrow$ R4_gbi
 R4_d1340_gbi: R4_d1340, 75 km_R4 $\Rightarrow$ R4_gbi
 R4_b1535_gbi: R4_b1535, 0 km_R4 $\Rightarrow$ R4_gbi
 R4_k1625_gbi: R4_k1625, 187 km_R4 $\Rightarrow$ R4_gbi
 R4_e1140_gbi: R4_e1140, 24 km_R4 $\Rightarrow$ R4_gbi
 P1_R5_b0800_f0915: R5_b0800, 74 km_R5 $\Rightarrow$ P1, R5_f0915
 P2_R5_t1030_f1125: R5_t1030, 54 km_R5 $\Rightarrow$ P2, R5_f1125
 P3_R5_k1100_d1340: R5_k1100, 162 km_R5 $\Rightarrow$ P3, R5_d1340
 P4_R5_s1430_b1535: R5_s1430, 68 km_R5 $\Rightarrow$ P4, R5_b1535
 P5_R5_t1350_k1625: R5_t1350, 155 km_R5 $\Rightarrow$ P5, R5_k1625
 P6_R5_v1000_e1140: R5_v1000, 101 km_R5 $\Rightarrow$ P6, R5_e1140
 R5_f0915_t1030: R5_f0915, 54 km_R5 $\Rightarrow$ R5_t1030
 R5_f0915_s1430: R5_f0915, 17 km_R5 $\Rightarrow$ R5_s1430
 R5_f0915_t1350: R5_f0915, 54 km_R5 $\Rightarrow$ R5_t1350
 R5_f0915_v1000: R5_f0915, 44 km_R5 $\Rightarrow$ R5_v1000
 R5_f1125_s1430: R5_f1125, 17 km_R5 $\Rightarrow$ R5_s1430
 R5_f1125_t1350: R5_f1125, 54 km_R5 $\Rightarrow$ R5_t1350
 R5_d1340_s1430: R5_d1340, 40 km_R5 $\Rightarrow$ R5_s1430
 R5_e1140_s1430: R5_e1140, 53 km_R5 $\Rightarrow$ R5_s1430
 R5_e1140_t1350: R5_e1140, 53 km_R5 $\Rightarrow$ R5_t1350
 R5_garage: R5_gki $\Rightarrow$ R5_gko
 R5_gko_b0800: R5_gko, 187 km_R5 $\Rightarrow$ R5_b0800
 R5_gko_t1030: R5_gko, 155 km_R5 $\Rightarrow$ R5_t1030
 R5_gko_k1100: R5_gko, 0 km_R5 $\Rightarrow$ R5_k1100
 R5_gko_s1430: R5_gko, 142 km_R5 $\Rightarrow$ R5_s1430
 R5_gko_t1350: R5_gko, 155 km_R5 $\Rightarrow$ R5_t1350
 R5_gko_v1000: R5_gko, 77 km_R5 $\Rightarrow$ R5_v1000
 R5_f0915_gki: R5_f0915, 125 km_R5 $\Rightarrow$ R5_gki
 R5_f1125_gki: R5_f1125, 125 km_R5 $\Rightarrow$ R5_gki
 R5_d1340_gki: R5_d1340, 162 km_R5 $\Rightarrow$ R5_gki
 R5_b1535_gki: R5_b1535, 187 km_R5 $\Rightarrow$ R5_gki

```

R5_k1625_gki: R5_k1625, 0 km_R5 => R5_gki
R5_e1140_gki: R5_e1140, 175 km_R5 => R5_gki
P1_R6_b0800_f0915: R6_b0800, 74 km_R6 => P1, R6_f0915
P2_R6_t1030_f1125: R6_t1030, 54 km_R6 => P2, R6_f1125
P3_R6_k1100_d1340: R6_k1100, 162 km_R6 => P3, R6_d1340
P4_R6_s1430_b1535: R6_s1430, 68 km_R6 => P4, R6_b1535
P5_R6_t1350_k1625: R6_t1350, 155 km_R6 => P5, R6_k1625
P6_R6_v1000_e1140: R6_v1000, 101 km_R6 => P6, R6_e1140
R6_f0915_t1030: R6_f0915, 54 km_R6 => R6_t1030
R6_f0915_s1430: R6_f0915, 17 km_R6 => R6_s1430
R6_f0915_t1350: R6_f0915, 54 km_R6 => R6_t1350
R6_f0915_v1000: R6_f0915, 44 km_R6 => R6_v1000
R6_f1125_s1430: R6_f1125, 17 km_R6 => R6_s1430
R6_f1125_t1350: R6_f1125, 54 km_R6 => R6_t1350
R6_d1340_s1430: R6_d1340, 40 km_R6 => R6_s1430
R6_e1140_s1430: R6_e1140, 53 km_R6 => R6_s1430
R6_e1140_t1350: R6_e1140, 53 km_R6 => R6_t1350
R6_garage: R6_gbi => R6_gbo
R6_gbo_b0800: R6_gbo, 0 km_R6 => R6_b0800
R6_gbo_t1030: R6_gbo, 54 km_R6 => R6_t1030
R6_gbo_k1100: R6_gbo, 187 km_R6 => R6_k1100
R6_gbo_s1430: R6_gbo, 68 km_R6 => R6_s1430
R6_gbo_t1350: R6_gbo, 54 km_R6 => R6_t1350
R6_gbo_v1000: R6_gbo, 120 km_R6 => R6_v1000
R6_f0915_gbi: R6_f0915, 74 km_R6 => R6_gbi
R6_f1125_gbi: R6_f1125, 74 km_R6 => R6_gbi
R6_d1340_gbi: R6_d1340, 75 km_R6 => R6_gbi
R6_b1535_gbi: R6_b1535, 0 km_R6 => R6_gbi
R6_k1625_gbi: R6_k1625, 187 km_R6 => R6_gbi
R6_e1140_gbi: R6_e1140, 24 km_R6 => R6_gbi

mutually_exclusive_sets_of_operating_units:
P1elo: P1_R1_b0800_f0915, P1_R2_b0800_f0915,
      P1_R3_b0800_f0915, P1_R4_b0800_f0915,
      P1_R5_b0800_f0915, P1_R6_b0800_f0915
P2elo: P2_R1_t1030_f1125, P2_R2_t1030_f1125,
      P2_R3_t1030_f1125, P2_R4_t1030_f1125,
      P2_R5_t1030_f1125, P2_R6_t1030_f1125
P3elo: P3_R1_k1100_d1340, P3_R2_k1100_d1340,
      P3_R3_k1100_d1340, P3_R4_k1100_d1340,
      P3_R5_k1100_d1340, P3_R6_k1100_d1340
P4elo: P4_R1_s1430_b1535, P4_R2_s1430_b1535,
      P4_R3_s1430_b1535, P4_R4_s1430_b1535,
      P4_R5_s1430_b1535, P4_R6_s1430_b1535
P5elo: P5_R1_t1350_k1625, P5_R2_t1350_k1625,
      P5_R3_t1350_k1625, P5_R4_t1350_k1625,
      P5_R5_t1350_k1625, P5_R6_t1350_k1625
P6elo: P6_R1_v1000_e1140, P6_R2_v1000_e1140,
      P6_R3_v1000_e1140, P6_R4_v1000_e1140,
      P6_R5_v1000_e1140, P6_R6_v1000_e1140

```

```

R1ki: R1_gvo_b0800, R1_gvo_t1030, R1_gvo_k1100,
      R1_gvo_s1430, R1_gvo_t1350, R1_gvo_v1000
R2ki: R2_gdo_b0800, R2_gdo_t1030, R2_gdo_k1100,
      R2_gdo_s1430, R2_gdo_t1350, R2_gdo_v1000
R3ki: R3_gto_b0800, R3_gto_t1030, R3_gto_k1100,
      R3_gto_s1430, R3_gto_t1350, R3_gto_v1000
R4ki: R4_gbo_b0800, R4_gbo_t1030, R4_gbo_k1100,
      R4_gbo_s1430, R4_gbo_t1350, R4_gbo_v1000
R5ki: R5_gko_b0800, R5_gko_t1030, R5_gko_k1100,
      R5_gko_s1430, R5_gko_t1350, R5_gko_v1000
R6ki: R6_gbo_b0800, R6_gbo_t1030, R6_gbo_k1100,
      R6_gbo_s1430, R6_gbo_t1350, R6_gbo_v1000
R1be: R1_f0915_gvi, R1_f1125_gvi, R1_d1340_gvi,
      R1_b1535_gvi, R1_k1625_gvi, R1_e1140_gvi
R2be: R2_f0915_gdi, R2_f1125_gdi, R2_d1340_gdi,
      R2_b1535_gdi, R2_k1625_gdi, R2_e1140_gdi
R3be: R3_f0915_gti, R3_f1125_gti, R3_d1340_gti,
      R3_b1535_gti, R3_k1625_gti, R3_e1140_gti
R4be: R4_f0915_gbi, R4_f1125_gbi, R4_d1340_gbi,
      R4_b1535_gbi, R4_k1625_gbi, R4_e1140_gbi
R5be: R5_f0915_gki, R5_f1125_gki, R5_d1340_gki,
      R5_b1535_gki, R5_k1625_gki, R5_e1140_gki
R6be: R6_f0915_gbi, R6_f1125_gbi, R6_d1340_gbi,
      R6_b1535_gbi, R6_k1625_gbi, R6_e1140_gbi
R1_f0915mx: R1_f0915_t1030, R1_f0915_s1430,
            R1_f0915_t1350, R1_f0915_v1000, R1_f0915_gvi
R1_f1125mx: R1_f1125_s1430, R1_f1125_t1350, R1_f1125_gvi
R1_e1140mx: R1_e1140_s1430, R1_e1140_t1350, R1_e1140_gvi
R2_f0915mx: R2_f0915_t1030, R2_f0915_s1430,
            R2_f0915_t1350, R2_f0915_v1000, R2_f0915_gdi
R2_f1125mx: R2_f1125_s1430, R2_f1125_t1350, R2_f1125_gdi
R2_e1140mx: R2_e1140_s1430, R2_e1140_t1350, R2_e1140_gdi
R3_f0915mx: R3_f0915_t1030, R3_f0915_s1430,
            R3_f0915_t1350, R3_f0915_v1000, R3_f0915_gti
R3_f1125mx: R3_f1125_s1430, R3_f1125_t1350, R3_f1125_gti
R3_e1140mx: R3_e1140_s1430, R3_e1140_t1350, R3_e1140_gti
R4_f0915mx: R4_f0915_t1030, R4_f0915_s1430,
            R4_f0915_t1350, R4_f0915_v1000, R4_f0915_gbi
R4_f1125mx: R4_f1125_s1430, R4_f1125_t1350, R4_f1125_gbi
R4_e1140mx: R4_e1140_s1430, R4_e1140_t1350, R4_e1140_gbi
R5_f0915mx: R5_f0915_t1030, R5_f0915_s1430,
            R5_f0915_t1350, R5_f0915_v1000, R5_f0915_gki
R5_f1125mx: R5_f1125_s1430, R5_f1125_t1350, R5_f1125_gki
R5_e1140mx: R5_e1140_s1430, R5_e1140_t1350, R5_e1140_gki
R6_f0915mx: R6_f0915_t1030, R6_f0915_s1430,
            R6_f0915_t1350, R6_f0915_v1000, R6_f0915_gbi
R6_f1125mx: R6_f1125_s1430, R6_f1125_t1350, R6_f1125_gbi
R6_e1140mx: R6_e1140_s1430, R6_e1140_t1350, R6_e1140_gbi

```

sources/truck_6_EUR_CO2_mx2.in