Tartalomjegyzék

1. fejezet

Bevezetés

Disszertációmban összefoglalom az elmúlt időszakban végzett kutatási tevékenységem, annak módszereit és eredményeit. Az első, Pluhár Andrással közös kutatási témám a perfekt gráfokkal volt kapcsolatos. A kilencvenes évek elején sokat olvastam speciális perfekt gráf osztályok strukturális jellemzéséről, tiltott részgráfokkal vagy egyéb módon. Négy hosszú kört, vagy annak komplementerét feszítve nem tartalmazó gráfoknak egy jellemzését sikerült adnunk. A [3] cikkben jelent meg közös munkánk eredménye. Ennek a disszertációnak ez a munka nem képezi tárgyát.

1.1. PNS kutatások

Imreh Balázs, korábbi operációkutatás tanárom, kollégám 1994-ben egy új, gyorsan fejlődő területtel ismertetett meg, ez volt a Process Network Synthesis (PNS) témaköre. A folyamatszintézis PNS modellben történő megfogalmazása, vizsgálata, az első eredmények Friedler Ferenc nevéhez, munkásságához fűződnek. K. Tarján, L. T. Fan, Y. W. Huang, valamint Friedler Ferenc és veszprémi kollégái alapozták meg e kutatási területet [18], [19], [20], [21], [22], [23]. Szegedről Imreh Balázs már a kezdetek után bekapcsolódott a PNS optimalizálási kérdéseinek megfogalmazásába és a strukturális modell kombinatorikus kérdéseinek vizsgálatába [25], [34], [35]. A PNS problémákat vizsgáló csoportjában, annak létrejötte óta együtt dolgoztunk Kovács Zoltán kollégámmal, majd később Imreh Csanád és Holló Csaba is részt vett a munkában. Több előadásban ismertettük eredményeinket nemzetközi kon-

ferenciákon. Többek között az alábbi cikkek születtek ebből az időszakból: [4], [5], [6], [7], [8], [9], [10]. A PNS témában tehát témavezetőmnek Imreh Balázs tekinthető.

A folyamatszintézist talán leginkább vegyipari folyamatok ihlették. Azonban a létrehozott modell sokkal több gyakorlati kérdéshez kapcsolódik, mindig felmerül ez a szemléletmód, tárgyalási módszertan, amikor valamilyen absztrakt anyaghalmazból egy másikat létre kell hozni. Ha a gyártás bizonyos műveleti egységekkel lehetséges, amelyekről mindössze annyit tudunk, hogy milyen anyaghalmazból milyet állítanak elő, akkor máris a PNS modell juthat eszünkbe. Nemcsak az ipari termeléssel kapcsolatban, hanem bizonyos gazdasági folyamatok, pénzügyi tranzakciók, társadalmi folyamatok vagy éppen az Interneten történő információáramlás bizonyos kérdéseinek matematikai megfogalmazásai is elvezethetnek a Process Network Synthesis valamelyik modelljéhez.

Az első hozzájárulásom a PNS elméletéhez a leginkább vizsgált kombinatorikus optimalizálási modell bonyolultságának megállapítása volt. Ha a műveleti egységek költségeinek összegét tekintjük minimalizálandónak, akkor a feladat NP-nehéz [4]. Ez az észrevétel megnyitotta annak lehetőségét, hogy nem hatékony megoldó eljárásokat, ilyenek fejlesztését is értékesnek tekintsük. Másrészt gyors, az optimumot közelítő heurisztikus algoritmusok is felértékelődtek a probléma bonyolultságának ismeretében. A jól- megoldható osztályok keresése, kutatása is fontos irány lett [9], [10].

Az optimalizálási feladatnak a korlátozás és szétválasztás kereteljárásban történő megoldásakor bevezették a döntési leképezés fogalmát [19], [21], [35]. Kiderült, hogy az ún. konzisztens döntési leképezések szoros kapcsolatban vannak a lehetséges megoldásokkal, számukat tehát fontos felülről becsülni. Az [5] munkában sikerült egységes elméleti becslést adni erre. Ezen értékek formulával való megadása általában nem lehetséges, bizonyos speciális tulajdonságok esetén azonban igen. Az ún. szeparátor típusú gépek esetén már adható volt formula, kiszámítása azonban még mindig nehéz. Sikerült olyan PNS probléma osztályokat definiálni, amelyek esetében már ki tudtam számolni a korlátokat [6], [7].

A [4], [5], [6], [7] közös cikkeink eredményeit szerzőtársaimmal oszthatatlannak tekintjük.

Lehet egy PNS modell azért könnyebben megoldható, mert a P-gráf nem teljesen általános, de lehet azért is, mert a célfüggvényünk egyszerűbb. Ebbe az irányba tettem lépéseket a [8] cikkel. Ha egy működő rendszer megépítésében az a fő szempont, hogy annak legköltségesebb eleme legyen minél olcsóbb, akkor a PNS Bottleneck változata jut eszünkbe. Ez éppen a célfüggvényben tér el lényegesen a Minsum változattól. Kiderült [8], hogy a Bottleneck változat hatékonyan megoldható. Ennek általánosítása, viszont a Minsum esetnek lényegében korlátozott verziója, a PNS k-összeg modellje. Erre is adtunk megoldást, amely rögzített k-ra hatékony.

Kiderült [4], hogy a PNS szoros kapcsolatban van halmazlefedési problémákkal. Később az is világossá vált [24], [37], hogy kölcsönösen speciális esetei egymásnak. Éppen ezért olyan heurisztikus megoldási módszert akartam megadni, amely legalább szakaszosan optimálisan próbálja előállítani az aktuálisan szükséges anyaghalmazt. Ráadásul egy-egy lépésben hatékony, mély matematikai hátterű algoritmus fut. Edmonds híres kehely-típusú algoritmusával rokon a minimális költségű éllefedési algoritmus (MCEC). Keserű Kornél programozói szakdolgozatának témája ennek a kevéssé alkalmazott algoritmusnak a megvalósítása volt, díjazott OTDK munkát is készített belőle. Az MCEC segédeljárásra építve definiáltam több, szakaszosan optimális döntéseket hozó heurisztikus megoldót ún., egyszerűsített PNS problémákra [11], [12]. Ez utóbbi fogalom nem speciális PNS feladatosztályt jelent, mert minden P-gráf átalakítható ekvivalens, egyszerűsített P-gráfra. Az algoritmusok gyorsan lefutottak, és az optimálishoz közeli lehetséges megoldásokat adtak.

1.2. Tökéletes dominancia

Egy számítógéphálózatban néhány helyre van mód nagy teljesítményű, komoly számításokat végző gépeket telepíteni, míg más csomópontokban a kiszolgálók, adatbázisok foglalnak helyet. Nyilván két tényező fontos, amelyek egymás ellen hatnak. Egyrészt minden kiszolgálóhoz legyen elég közel szerver, másrészt ne kelljen sok erős gépet felhasználni. Keresendőek tehát az irányított, vagy irányítatlan hálózatban azok a csomópontok, amelyektől minden más csúcs legfeljebb d lépésben elérhető. Kolozsvári kollégámmal, Kása Zoltánnal kezdtünk gondolkozni ilyen jellegű alapkérdéseken, ő ajánlotta figyelmembe a [42] cikket, melyben a szerzők eltérő hálózati topoló-

giákat vizsgáltak éppen ebből a szempontból. Ebben a de Bruijn gráfokkal kapcsolatban mind az irányított, mind az irányítatlan esetben, bizonyos paraméterekre megoldatlan kérdéseket fogalmaztak meg. Sikerült választ adni [13] két nyitott kérdésre perfekt dominál halmazok létezésével kapcsolatban.

1.3. Belső szállítások

A gyakorlati optimalizálás motiválta a dolgozat harmadik témáját is. Logisztikai témájú kutatásaink közben szinte minden kérdés eltér egy kicsit a híres, sokat kutatott alapkérdések mindegyikétől. Vannak azonban kapcsolódási pontok, felhasználhatjuk az elméleti, vagy gyakorlati informatika eredményeit, így tudunk eredményesen optimalizálni. Az optimális fuvarszervezési problémákon gondolkozva egy közös általánosítását fogalmaztam meg a TSP és LOP alapproblémáknak. Ha az utazó ügynök olcsóbban szeretné körbejárni a klienseit, de az útja során az ügyfelek között bizonyos tranzakciókat szeretne lebonyolítani, amely neki hasznot hoz, akkor kettős célt tűz ki maga elé, amelyek egymást zavarják. Ha az egyik heurisztikus elképzelés logikus az egyik nehéz problémára, egy másik pedig a másikra, akkor vajon hogyan kell eljárni ha mindkét cél, illetve ezek eredője a fontos?

A HPPIT problémára [14] megoldó algoritmusokat definiáltam, szerzőtársaim is továbbiakat, majd a megvalósítás után Bartók Tamás OTDK díjas munkájában értékelte a futási tapasztalatokat. A feladatok definiálásakor fontos volt az adatokat úgy megadni, hogy igazi HPPIT problémákat oldjunk meg, ne domináljon sem a TSP, sem a LOP célfüggvényrész.

A dolgozatban bemutatott eredmények legnagyobb részét a [4], [5], [6], [8], [11], [12], [13] és [14] cikkekben publikáltam.

1.4. Köszönetnyilvánítás

Legelőször Imreh Balázsnak, tanáromnak, szeretett kollégámnak köszönöm a szakmai útmutatását, a munkám hátterének biztosítását, a konferenciák lehetőségét. Emberségét mint vezető, inspirációját mint témavezető, megértő türelmét jelen munka elkészültének halogatásáért. Köszönöm, hogy barátjának fogadott. Kitölthetetlen űrt hagyott a szívemben; ma már nem lehet közöttünk.

Köszönöm szerzőtársaimnak, kollégáimnak, tanítványaimnak a közös munkát, az ötleteket, a megfogalmazásokat, a technikai segítséget.

A disszertáció alapjául szolgáló előadásokat, cikkeket a következő pénzügyi alapok támogatták: MKM 635/96; FKFP 0008/1999; OTKA T030074; NKFP-2/015/2004; OTKA T046405.

Köszönöm a segítségét mindazon régi és jelenlegi kollégáimnak, akikhez mindig fordulhattam jó tanácsért, akik készségesek voltak és segítettek megoldani kisebb-nagyobb gondjaimat.

Egykori iskoláimban, az egyetemen mindig voltak olyan kiemelkedő tanár egyéniségek, akik meghatározó hatással voltak rám. Nekik köszönhetem, hogy a matematikát választottam, hogy a természettudományokhoz, informatikához vonzódtam. Régi osztálytársaimnak, évfolyamtársaimnak is köszönöm, hogy sokat tanulhattam a beszélgetéseinkből.

Családom szeretete, támogatása, buzdítása nélkül nem tudtam volna elkészíteni a disszertációt és az annak alapjául szolgáló kutatásokat.

Ajánlom ezt a munkát Drága Szüleimnek, akik lehetővé tették az utat, amely idáig elvezetett.

2. fejezet

A PNS modell alapfogalmai

2.1. A PNS probléma

Amíg a címben álló fogalom világosabbá nem válik, sok definícióra, megállapításra lesz szükségünk. Már itt elöljáróban fontos megjegyezni, hogy a dolgozat különböző részeiben a PNS probléma kifejezés eltérő jelentéssel is bírhat. A jelen és a 3., 4. fejezetben áttekintést adunk a PNS probléma alapfogalmairól. A bemutatott definíciók, korábbi eredmények, egyrészt szükségesek a tárgyalt eredményekben szereplő fogalmak megértéséhez, másrészt bővebben megtalálhatók a [5], [6], [18], [19], [20], [21], [22], [23], [24], [25], [32], [34], [35], [37], [40] munkákban.

Jelölje $\varphi(H)$ egy tetszőleges H halmaz összes részhalmazát, $\varphi'(H)$ pedig a H halmaz összes nemüres részhalmazát. Legyenek M és $O \subseteq \varphi'(M) \times \varphi'(M)$ véges, nemüres, és diszjunkt halmazok. Az M elemei az anyagok, míg az O elemei a műveleti egységek, melyek segítségével bizonyos bemenő anyagokból nyerünk előírt módon egy kimeneti anyaghalmazt. Hogy mi megy végbe a műveleti egységekben, azzal nem foglalkozunk. Figyelmen kívül hagyjuk továbbá azt is, hogy miként kezdett a rendszer működni, csak statikus termeléssel foglalkozunk. Formálisan bármely $u \in O$ műveleti egységre $u = (\alpha, \beta)$, ahol α a bemeneti (nemüres) anyaghalmaz, β pedig a kimeneti (nem üres) anyaghalmaz. Azt fogjuk mondani, hogy az u műveleti egység az α anyaghalmazból a β anyaghalmazt gyártja.

2.1.1. Definíció. *Az (M, O) párhoz egyértelműen hozzárendelhető egy irányí-*

tott, kétrészes gráf, amit a ***folyamat gráfjának*** *nevezünk:*
$PG(M,O) = (M \cup O, A_1 \cup A_2)$*, ahol az élhalmaz kétféle típusú élből áll:*

$$A_1 = \{(X,Y) : Y = (\alpha,\beta) \in O \text{ és } X \in \alpha\},$$
$$A_2 = \{(Y,X) : Y = (\alpha,\beta) \in O \text{ és } X \in \beta\}.$$

2.1.2. Definíció. *Egy* (V', E') *gráf egy* $PG(M,O)$ *folyamat gráf* ***részgráfja****, ha:*

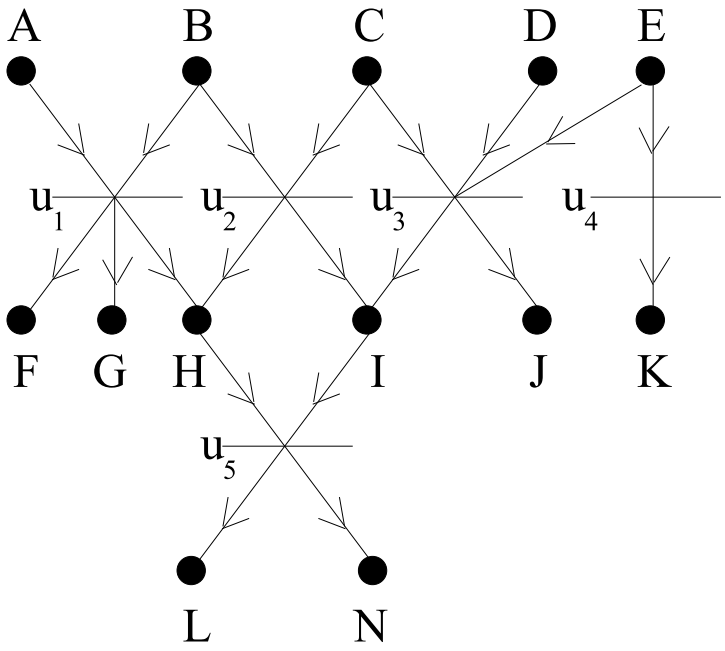
- $V' = M' \cup O',\ M' \subseteq M,\ O' \subseteq O,$
- $O' \subseteq \varphi'(M') \times \varphi'(M'),$
- $E' = A'_1 \cup A'_2$*, ahol*

$$A'_1 = \{(X,Y) : Y = (\alpha,\beta) \in O' \text{ és } X \in \alpha\},$$
$$A'_2 = \{(Y,X) : Y = (\alpha,\beta) \in O' \text{ és } X \in \beta\}.$$

2.1.1. Megjegyzés. Vegyük észre, hogy adott (M,O) pár és a hozzárendelt folyamat gráf kölcsönösen egyértelműen meghatározzák egymást. Ezért a továbbiakban az (M,O) párokat azonosítani fogjuk a hozzájuk rendelt folyamat gráfokkal.

Egy $X \in M$ anyag ***forrás*** (M,O)-ban, ha nem létezik (Y,X) él a folyamat gráfban. Ha léteznek $X_1, X_2, ..., X_n$ csúcspontok a gráfban, melyekre $(X_1,X_2),(X_2,X_3),\ldots,(X_{n-1},X_n)$ élek az (M,O) folyamat gráfban, akkor az ezen csúcspontok által meghatározott ***utat*** $[X_1,X_n]$-el fogjuk jelölni. Mivel (M,O) páros gráf, ezért bármely $[X_1,X_n]$ útban felváltva következnek anyagi, illetve műveleti egység típusú csúcsok.

Legyen most $P \subseteq M$ és $R \subseteq M$ az ***előállítandó anyagok*** és a ***felhasználható nyersanyagok*** egymástól diszjunkt halmaza. Az előállítandó anyagokra szinonímaként fogjuk használni a ***céltermék*** vagy ***kívánt anyag*** kifejezéseket, a felhasználható nyersanyagokat pedig egyszerűen ***nyersanyagoknak*** nevezzük. Ha a műveleti egységek halmaza O, akkor jelölje M' azon anyagok halmazát, amelyek O-beli gépek bemeneti vagy kimeneti anyagai. Legyen $M = M' \cup P \cup R$. Ekkor (M,O) a folyamat gráfja. Az $\mathbf{M}=(P,R,O)$ hármast pedig a tekintett PNS-probléma ***strukturális modelljének*** nevezzük.



2.1. ábra.

2.1.1. Példa. *Legyen* $\mathbf{M}=(P,R,O)$*, melyben*

- $P=\{L,N\}$,
- $R=\{A,B,C,D,E\}$,
- $O=\{u_1,u_2,u_3,u_4,u_5\}$, *ahol*
  - $u_1=(\{A,B\},\{F,G,H\})$,
  - $u_2=(\{B,C\},\{H,I\})$,
  - $u_3=(\{C,D,E\},\{I,J\})$,
  - $u_4=(\{E\},\{K\})$, *és*
  - $u_5=(\{H,I\},\{L,N\})$.

Ekkor $M=\{A,B,C,D,E,F,G,H,I,J,K,L,N\}$ *és az* (M,O) *folyamat gráfot a 2.1. ábra szemlélteti.*

2.1.3. Definíció. *Legyen adott egy* $\mathbf{M} = (\mathbf{P}, \mathbf{R}, \mathbf{O})$ *strukturális modell. Ekkor egy* (m, o) *részgráfot az* $\mathbf{M}$ *strukturális modell egy* ***lehetséges megoldás struktúrájának*** *nevezünk, ha teljesülnek a következő tulajdonságok:*

($\mathcal{A}1$) $P \subseteq m$,

($\mathcal{A}2$) $\forall X \in m$, $X \in R \Leftrightarrow$ *nem létezik* (Y, X) *él* (m, o)*-ban,*

($\mathcal{A}3$) $\forall Y_0 \in o$, $\exists\, [Y_o, Y_n]$ *út, amelyre* $Y_n \in P$,

($\mathcal{A}4$) $\forall X \in m$, $\exists (\alpha, \beta) \in o$ *úgy, hogy* $X \in \alpha \cup \beta$.

Jelölje $S(\mathbf{M})$ az $\mathbf{M}$ strukturális modell lehetséges megoldás struktúráinak halmazát. Egy lehetséges megoldás struktúrát tehát úgy képzelhetünk el, mint a folyamat gráfjának egy olyan részhálózatát, melyben

- szerepelnek az előállítandó anyagok,
- nyersanyagokat nem gyártunk, és minden nem nyersanyagot gyártja valamelyik műveleti egységünk,
- csak olyan műveleti egységet működtetünk, amely legalább közvetve részt vesz valamelyik előállítandó anyag gyártásában, illetve
- nincs izolált anyagi pont a részgráfban.

2.1.2. Példa. *Tekintsük a 2.1.1 példát. Akkor*

- $(m_1, o_1) = (\{A, B, F, G, H, C, D, E, I, J, L, N\}, \{u_1, u_3, u_5\})$,
- $(m_2, o_2) = (\{B, C, H, I, L, N\}, \{u_2, u_5\})$
- $(m_3, o_3) = (\{A, B, F, G, H, C, I, L, N\}, \{u_1, u_2, u_5\})$,
- $(m_4, o_4) = (\{B, C, D, E, H, I, J, L, N\}, \{u_2, u_3, u_5\})$
- $(m_5, o_5) = (\{A, B, C, D, E, F, G, H, I, J, L, N\}, \{u_1, u_2, u_3, u_5\})$

az összes lehetséges megoldás struktúrák, míg például

- $(m_6, o_6) = (\{C, D, E, H, I, J, L, N\}, \{u_3, u_5\})$ *és*
- $(m_7, o_7) = (\{B, C, H, I, E, K, L, N\}, \{u_2, u_4, u_5\})$

nem lehetséges megoldás struktúrák az ($\mathcal{A}2$)*, illetve az* ($\mathcal{A}3$) *feltételek nem teljesülése miatt.*

2.1.2. Megjegyzés. Az M és O halmazok végessége miatt $\mathrm{S}(\mathbf{M})$ is véges halmaz.

2.1.3. Megjegyzés. Általános esetben az ($\mathcal{A}1$) – ($\mathcal{A}4$) feltételeket teljesítő lehetséges megoldás struktúrák halmaza üres halmaz is lehet: a 2.1.1. példában, ha a B anyag nem lenne nyersanyag, akkor $\mathrm{S}(\mathbf{M}) = \emptyset$ -t kapnánk.

2.1.1. Lemma. ([22]) *Legyen* $\mathbf{M}$=(P, R, O) *egy PNS probléma strukturális modellje. Ha* (m, o) *és* (m', o') *lehetséges megoldás struktúrái* $\mathbf{M}$*-nek, akkor* $(m, o) \cup (m', o')$ *is lehetséges megoldás struktúrája* $\mathbf{M}$*-nek.*

2.1.1. Következmény. *A 2.1.2. megjegyzés és 2.1.1. lemma alapján* $\mathbf{S(M)}$ *összes lehetséges megoldás struktúráinak egyesítése is lehetséges megoldás struktúra lesz.*

2.1.4. Definíció. *Legyen* $\mathbf{M}$=(P, R, O) *egy PNS probléma strukturális modellje.* $\mathbf{M}$ ***maximális struktúrája*** *alatt a*

$$\mu(\mathbf{M}) = \bigcup_{(m,o)\in\mathbf{S}(\mathbf{M})} (m, o)$$

megoldás struktúrát értjük. Ha $\mathbf{S(M)} = \emptyset$*, akkor* $\mu(\mathbf{M}) = \emptyset$ *és* $\mu(\mathbf{M})$ *-et* ***degenerált****nak nevezzük.*

2.1.4. Megjegyzés. $\mathrm{S}(\mathbf{M}) \neq \emptyset$ akkor és csakis akkor, ha $\mu(\mathbf{M}) \neq \emptyset$.

2.1.5. Definíció. *Legyen* $o \subseteq O$ *műveleti egységek egy halmaza. Definiáljuk a* mat^{in}*,* mat^{out}*, és mat függvényeket a következőképpen:*

$$mat^{in}(o) : \ \varphi'(O) \to \varphi'(M), \ mat^{in}(o) \ = \bigcup_{(\alpha,\beta)\in o} \alpha,$$

$$mat^{out}(o): \ \varphi'(O) \to \varphi'(M), \ mat^{out}(o) \ = \bigcup_{(\alpha,\beta)\in o} \beta,$$

és

$$mat(o): \ \varphi'(O) \to \varphi'(M), \ mat(o) \ = \ mat^{in}(o) \cup mat^{out}(o).$$

Szemléletesen a műveleti egységek egy o halmazához tartozó input anyagok, illetve output anyagok egyesítéséről van szó.

2.1.2. Lemma. ([22]) *Legyen* $\mathbf{M}=(P, R, O)$ *egy PNS probléma strukturális modellje. Ha* $(m, o) \in S(\mathbf{M})$*, akkor* $m = mat(o)$.

2.1.2. Következmény. ([22]) *Egy* (m, o) *lehetséges megoldás struktúra egyértelműen meghatározott az o műveleti egység halmazzal.*

2.2. A PNS probléma redukciója

Az eddigiekből nyilvánvaló, hogy egy $\mathbf{M}=(P, R, O)$ strukturális modell egyértelműen meghatározza az S($\mathbf{M}$) lehetséges megoldás struktúra halmazt. Ez fordítva azonban nem igaz: különböző strukturális modellek rendelkezhetnek azonos megoldás struktúra halmazzal. Például ha egy modellhez felveszünk olyan további műveleti egységeket, melyeknek bemeneti és kimeneti anyaghalmazaik diszjunktak az eredeti modell anyaghalmazaitól, akkor az így kapott strukturális modell ugyanazzal a megoldás struktúra halmazzal fog rendelkezni. Gyakorlati szempontból fontos lenne tehát megtalálni azt a legkisebb méretű, és így legkönnyebben kezelhető megoldás struktúrát, mely tartalmazza az eredeti probléma összes lehetséges megoldás struktúráját.

2.2.1. Definíció. *Jelöljük* $\mathcal{M}$*-el a PNS problémák strukturális modelljeinek halmazát. Azt mondjuk, hogy az* $\overline{\mathbf{M}} = \left(\overline{P}, \overline{R}, \overline{O}\right) \in \mathcal{M}$*, és az* $\mathbf{M'} = (P', R', O') \in \mathcal{M}$ *strukturális modellek* ***ekvivalensek****, és ezt* $\overline{\mathbf{M}} \sim \mathbf{M}'$ *-vel jelöljük, ha* $\overline{P} = P'$ *és* $S(\overline{\mathbf{M}}) = S(\mathbf{M}')$.

A $\sim$ reláció reflexív, tranzitív és szimmetrikus, azaz ekvivalencia reláció. Mivel gyakorlatilag csak az $\mathcal{M}$ nem üres megoldás struktúra halmazzal rendelkező ekvivalencia osztályai érdekesek, ezért a továbbiakban ilyen osztályokat fogunk tanulmányozni. Tetszőleges G' ilyen osztályra és $\mathbf{M} = (P, R, O) \in G'$, illetve $\mathbf{M'} = (P', R', O') \in G'$ strukturális modellekre definiáljuk a $\lhd$ relációt a következőképpen: $\mathbf{M} \lhd \mathbf{M'}$ akkor és csakis akkor, ha $(M, O) \subseteq (M', O')$ és $R \subseteq R'$, ahol (M, O) és (M', O') az $\mathbf{M}$, illetve $\mathbf{M'}$ strukturális modellek folyamat gráfjai. Nyilvánvalóan a $\lhd$ reláció reflexív, antiszimmetrikus és tranzitív, azaz részben rendezés G'-n.

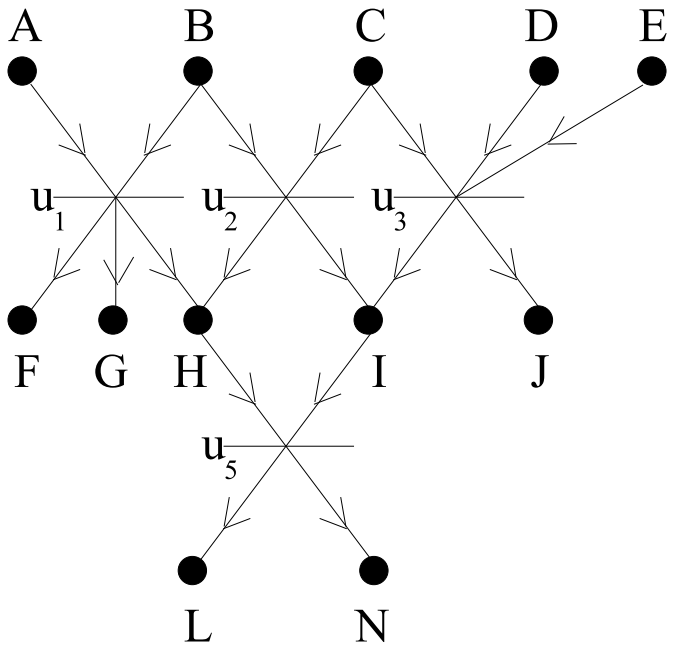
2.2.1. Példa. *Legyen* $\mathbf{M'}$=(P', R', O')*, melyben*

- $P' = \{L, N\}$,
- $R = \{A, B, C, D, E\}$,
- $O' = \{u_1, u_2, u_3, u_5\}$, *ahol*
  - $u_1 = (\{A, B\}, \{F, G, H\})$,
  - $u_2 = (\{B, C\}, \{H, I\})$,
  - $u_3 = (\{C, D, E\}, \{I, J\})$, *és*
  - $u_5 = (\{H, I\}, \{L, N\})$.

A folyamat gráfját a 2.2 ábra szemlélteti. Könnyen ellenőrizhető, hogy $P' = P$, $R' = R$, $O' \subseteq O$, *továbbá a folyamatok lehetséges megoldás struktúrái azonosak a 2.1.2. példában leírtakkal. Mindebből az következik, hogy* $\mathbf{M'} \lhd \mathbf{M}$. *Érdemes továbbá megfigyelni azt is, hogy a példák maximális struktúrái is azonosak:* $\mu(\mathbf{M}) = \mu(\mathbf{M'}) = (m_5, o_5)$.

Szeretnénk meghatározni adott ekvivalencia osztály egy minimális elemét a $\lhd$ részben rendezésre nézve. Legyen G' egy nem üres lehetséges megoldás struktúra halmazzal rendelkező ekvivalencia osztály. Legyen $\mathbf{M} = (P, R, O) \in G'$. Mivel $S(\mathbf{M}) \neq \emptyset$, ezért a 2.1.4. megjegyzés alapján $\mu(\mathbf{M})$ nem degenerált. Akkor $\mu(\mathbf{M}) = (\overline{M}, \overline{O})$-ra képezzük az $\overline{\mathbf{M}} = (P, \overline{R}, \overline{O})$ hármast, ahol $\overline{R} = R \cap \overline{M}$.

2.2.1. Lemma. *A fenti módon meghatározott* $\overline{\mathbf{M}} = (P, \overline{R}, \overline{O})$ *egy PNS probléma olyan strukturális modellje, mely ekvivalens* $\mathbf{M}$*-el és* $\overline{\mathbf{M}} \lhd \mathbf{M}$.



2.2. ábra. A 2.2.1. példához tartozó folyamat gráf.

2.2.1. Következmény. ([22]) *Minden* $M \in G'$*-re* $\overline{\mathbf{M}} \triangleleft \mathbf{M}$ *teljesül, ami azt jelenti, hogy* $\overline{\mathbf{M}}$ *az ekvivalencia osztály legkisebb eleme.*

Az ekvivalencia osztály legkisebb eleme a folyamat gráf olyan minimális részgráfjának felel meg, mely tartalmazza az összes lehetséges megoldás struktúrát, de nem tartalmaz olyan hálózati részeket, amelyek nem szükségesek egyik lehetséges megoldáshoz sem. Ennek megtalálása azért hasznos, mert csökkenti a probléma méretét.

2.2.2. Definíció. *Egy PNS probléma* $\mathbf{M} = (P, R, O)$ *strukturális modelljét a tekintett PNS probléma* ***redukált strukturális modelljének*** *nevezzük, ha* $S(\mathbf{M}) \neq \emptyset$*, és bármely más* $\mathbf{M}' \sim \mathbf{M}$ *strukturális modellre* $\mathbf{M} \triangleleft \mathbf{M}'$*.*

Tehát egy G' ekvivalencia osztály egy nem üres lehetséges megoldás struktúra halmazzal rendelkező $\mathbf{M} = (P, R, O)$ strukturális modelljéből kiindulva, a $\mu(\mathbf{M}) = (\overline{M}, \overline{O})$ maximális struktúra ismeretében, az $\overline{\mathbf{M}}$ redukált struktúrát az $\overline{\mathbf{M}} = (P, \overline{R}, \overline{O})$ módon kaphatjuk meg, ahol $\overline{R} = R \cap \overline{M}$.

Ehhez azonban tetszőleges strukturális modellről el kellene tudnunk dönteni, hogy a lehetséges megoldás struktúrák halmaza üres halmaz-e, és ha nem, akkor meg kellene tudnunk határozni a maximális struktúrát. Ennek megvalósítását fogjuk bemutatni a következő alfejezetben.

2.3. A maximális struktúra meghatározása

A vizsgálatok során bebizonyosodott, hogy a lehetséges megoldás struktúra halmaz ürességének eldöntésére és a maximális struktúra generálására adható egy hatékony, polinomiális idejű algoritmus ([18, 23]). További empirikus vizsgálatok megmutatták, hogy véletlenszerűen generált feladatokra a redukciós eljárás bizonyos, átlagosnak tekinthető feladatosztályok esetén közelítőleg 47%-ára csökkenti a feladat méretét ([35]). Ebben az alfejezetben a ([18])-ben leírt algoritmust fogjuk ismertetni.

Legyen $\mathbf{M} = (P, R, O)$ egy PNS probléma strukturális modellje.

Maximális struktúra generáló algoritmus (MSG)

1. Redukció

Inicializálás

1.1. Legyen $O_0 = O \setminus \{(\alpha, \beta) : (\alpha, \beta) \in O \ \& \ \beta \cap R \neq \emptyset\}$ és $M_0 = mat(O_0)$.

1.2. Ha $P \not\subseteq M_0$, akkor nem létezik **M**-re maximális struktúra és az algoritmus véget ér.

1.3. Legyen $T_0 = \{X : X \in M_0 \setminus R \ \& \ ((\alpha, \beta) \in O_0 \longrightarrow X \notin \beta)\}$.

1.4. Legyen $r = 0$.

Iteráció

1.5. Ha $T_r = \emptyset$, akkor folytassuk az eljárást a 2.1. lépéssel.

1.6. Válasszunk egy $X \in T_r$ anyagot.

1.7. Legyen $O_X = \{(\alpha, \beta) : (\alpha, \beta) \in O_r \ \& \ X \in \alpha\}$.

1.8. Legyen $O_{r+1} = O_r \setminus O_X$ és $M_{r+1} = mat(O_{r+1})$.

1.9. Ha $P \not\subseteq M_{r+1}$, akkor nem létezik **M**-re maximális struktúra és az algoritmus véget ér. Egyébként folytassuk az eljárást a következő lépéssel.

1.10. Határozzuk meg a

$$T'_r = \{Y : Y \in mat^{out}(O_X) \,\&\, Y \notin mat^{out}(O_{r+1}) \,\&\, Y \in mat^{in}(O_{r+1})\}$$

halmazt.

1.11. Legyen $T_{r+1} = (T_r \cap M_{r+1}) \cup T'_r$.

1.12. Növeljük eggyel az r iterációszámot.

1.13. Kezdjünk egy új iterációt az 1.5. lépéssel.

2. Építés

Inicializálás

2.1. Legyen $W_0 = P$, $m_0 = \emptyset$, $o_0 = \emptyset$ és $s = 0$.

Iteráció

2.2. Ha $W_s = \emptyset$, akkor vége: kaptunk egy megoldás struktúrát **M**-re.
Ha $\bar{m} = mat(o_s)$, akkor $(\bar{m}, o_s)$ az **M** maximális struktúrája.
Ha $W_s \neq \emptyset$, akkor folytassuk az eljárást a következő lépéssel.

2.3. Válasszunk egy tetszőleges X anyagot W_s-ből.

2.4. Legyen $m_{s+1} = m_s \cup \{X\}$.

2.5. Alkossuk meg az $O_X^* = \{(\alpha, \beta) : (\alpha, \beta) \in O_r \ \& \ X \in \beta\}$ halmazt.

2.6. Legyen $o_{s+1} = o_s \cup O_X^*$.

2.7. $W_{s+1} = (W_s \cup mat^{in}(O_X^*)) \setminus (R \cup m_{s+1})$.

2.8. Növeljük eggyel az s iterációszámot.

2.9. Kezdjünk egy új iterációt a 2.2. lépéssel.

Az algoritmus két fő részből áll. Az első, redukciós részben töröljük azokat a műveleti egységeket, melyek vagy nyersanyagot is termelnek, vagy valamely nyersanyagtól különböző, egyetlen műveleti egység által sem termelt bemeneti anyag hiányában nem tudnának működni. Ha közben azt tapasztaljuk, hogy valamely célterméket egyetlen megmaradt műveleti egységünk sem gyártja, ez azt jelenti, hogy nincs lehetséges megoldás struktúra, azaz $S(\mathbf{M}) = \emptyset$, de akkor maximális struktúra sincs, így az algoritmust megállíthatjuk.

A második részben a rendelkezésre álló műveleti egységekből felépítjük azt a hálózatot, mely kizárólag hasznos anyagokat termelő műveleti egységeket tartalmaz. Először kiindulunk abból, hogy a céltermékeket le kell gyártani. Ehhez minden olyan műveleti egység hasznos lehet, mely célterméket gyárt, tehát ezeket bevesszük a hálózatba. De a hálózatba bevett műveleti egységek bemeneti anyagait is le kellene gyártani, tehát bevesszük az azokat gyártó műveleti egységeket is, és így tovább.

Összesítve, az algoritmus eldönti, vajon az $S(\mathbf{M}) = \emptyset$ vagy létezik lehetséges megoldás struktúra. Az eredményként kapott hálózat azokat és csakis azokat a műveleti egységeket és anyagokat tartalmazza, amelyek résztvehetnek valamely lehetséges megoldás struktúrában, így a modell maximális struktúráját szolgáltatja. Lényeges továbbá az is, hogy az algoritmus polinomiális idő alatt oldja meg ezt a feladatot ([18]).

Az eddigiekben a PNS probléma azt jelentette, hogy adjunk válaszokat egy $\mathbf{M} = (P, R, O)$ strukturális modell megoldhatósági kérdéseire. A továbbiakban a PNS problémát optimalizálási problémává tesszük.

2.4. A súlyozott PNS modell

Legyen $\mathbf{M} = (P, R, O)$ egy PNS probléma strukturális modellje. Gyakorlati szempontból természetes igény, hogy szeretnénk meghatározni azt a lehetséges megoldás struktúrát, mely valamilyen szempontból a leggazdaságosabban állítja elő a céltermékeket. Definiálunk tehát a lehetséges megoldás struktúrák halmazán egy költségfüggvényt, és keresünk egy legkisebb költségű lehetséges megoldást. Legyen $z : S(\mathbf{M}) \to R_+$ egy ilyen költségfüggvény. Akkor a megoldandó feladat:

$$\min\{z((m,o)) \ : \ (m,o) \in S(\mathbf{M})\}. \tag{2.1}$$

Egy megoldás struktúra költségfüggvényét többféleképpen definiálhatjuk. Mi most ennek egy egyszerű és természetes definícióját fogjuk adni.

Legyen $w : O \to R_+$ egy költségfüggvény a műveleti egységeken. Egy lehetséges megoldás struktúra költségét a benne levő műveleti egységek összköltségeként fogjuk definiálni. Így a PNS probléma azon változatát vizsgáljuk, amikor a feladat egy olyan lehetséges megoldás struktúra meghatározása, melyre a benne működő műveleti egységek összsúlya minimális, vagyis

$$\min\{\sum_{u \in o} w(u) \ : \ (m,o) \in S(\mathbf{M})\}. \tag{2.2}$$

Az a kérdés, hogy meg tudjuk-e oldani ezt a feladatot, ha igen hogyan, és milyen hatékonysággal?

Mivel az M és O halmazok végessége miatt a lehetséges megoldás struktúrák száma is véges, továbbá adott lehetséges megoldás struktúra fentiekben definiált költségének kiszámítása is véges idő alatt elvégezhető, ezért nyilván a legkisebb költségű lehetséges megoldás struktúra meghatározása is véges időben megoldható feladat. Például egy lehetőség, hogy felsoroljuk a strukturális modellben szereplő O halmaz összes részhalmazát, mindegyikről megvizsgáljuk, hogy lehetséges megoldás struktúra-e, és a lehetséges megoldás struktúrák közül kiválasztjuk a legkisebb költségűt. Ez a módszer exponenciális. Túl sok részhalmazt vizsgálunk meg feleslegesen. Az előzőekben tárgyaltak ismeretében világos, hogy elegendő csak a maximális struktúra

műveleti egység részhalmazaival foglalkozni. A következő fejezetben látni fogjuk, hogy ennél hatékonyabb megoldást lehet ugyan találni, de sokkal hatékonyabb, polinomiális idejű megoldási módszer nem várható.

3. fejezet

A PNS probléma NP–nehéz

A [4] cikkben igazoltuk a fenti (2.2) probléma NP–nehézségét. A bizonyításunk alapötlete az volt, hogy megmutattuk a PNS problémák egy részosztályának ekvivalenciáját a halmazlefedési problémával, mely egy jól ismert NP–teljes probléma ([2, 41]).

3.1. A halmazlefedési feladat visszavezetése a PNS-re

Tekintsük a PNS problémák azon PNS_{w1} részosztályát, melyben $O \subseteq \varphi'(R) \times \varphi'(P)$, azaz melyben minden műveleti egység nyersanyagokból céltermékeket gyárt. Legyen tehát $O = \{u_1, \ldots, u_n\}$, $u_j = (\alpha_j, \beta_j) \in \varphi'(R) \times \varphi'(P)$, $j = 1, \ldots, n$. Ha a β_j halmazok súlyozását a következőképpen definiáljuk: $w'(\beta_j) = w(u_j)$, akkor könnyen belátható, hogy a PNS_{w1} probléma ekvivalens a P halmaz β_j halmazokkal való halmazlefedési problémájával.

Fordítva, tekintsünk egy tetszőleges halmazlefedési problémát. Ez azt jelenti, hogy egy tetszőleges nem üres, véges P halmazt le szeretnénk fedni súlyozott β_j, $j = 1, \ldots n$ halmazokkal, minél kisebb összköltséggel. Jelölje a halmazok súlyozását $w'(\beta_j)$, $j = 1, \ldots, n$. Legyen $R \neq \emptyset$ egy tetszőleges véges halmaz, melyre $R \cap P = \emptyset$. Legyenek továbbá $u_j = (R, \beta_j)$, $j = 1, \ldots, n$, és $O = \{u_1, \ldots, u_n\}$. Definiáljunk egy w súlyfüggvényt O-n a következőképpen: $w(u_j) = w'(\beta_j)$, $j = 1, \ldots, n$. Ha a P és R halmazokat anyaghalmazoknak, az O-t pedig műveleti egységek halmazának tekintjük,

akkor a (P, R, O) által meghatározott PNS_{w1} probléma ekvivalens a tekintett halmazlefedési problémával.

A fentiekből következik, hogy:

3.1.1. Lemma. ([4]) *A PNS_{w1} problémaosztály ekvivalens a halmazlefedési problémával.*

Mivel azonban a PNS_{w1} problémaosztály a PNS problémának egy részosztálya, a halmazlefedési probléma pedig NP–teljes, ezért a következő állítást kapjuk.

3.1.1. Tétel. ([4]) *A PNS probléma NP–nehéz.*

Egy NP–nehéz probléma NP–teljességéhez azt kellene megmutatni, hogy eleme az NP osztálynak. Ha veszünk egy feltételezett optimális megoldást, annak az ellenőrzése, hogy valóban megfelelő költségű lehetséges megoldás struktúra-e, polinomiális időben elvégezhető.

3.2. A PNS visszavezetése a halmazlefedési feladatra

A ([24, 37]) cikkekben a szerzők konstruktív módon fordított irányú visszavezetést adtak meg, melyet az alábbiakban vázolunk. Abból indulunk ki, hogy a lehetséges megoldás struktúrák mind a maximális struktúra részei, és ily módon, ha $(\overline{M}, \overline{O})$ az $\mathbf{M} = (P, R, O)$ strukturális modell maximális struktúrája, akkor az eredeti súlyozott PNS probléma ekvivalens a

$$\min \left\{ \sum_{u \in o} w(u) : \ (m, o) \in S(P, R \cap \overline{M}, \overline{O}) \right\} \tag{3.1}$$

feladattal.

Ha $S(P, R, O) = \emptyset$, akkor a PNS probléma ekvivalens azzal a halmazlefedési problémával, melyben a P halmazt kell lefedni annak egy valódi, tetszőleges súlyú részhalmazával: egyiknek sincs lehetséges megoldása.

Ha $S(P,R,O) \neq \emptyset$, akkor felhasználva a 2.1.2. következménybeli megállapítást, miszerint egy műveleti egység halmaz egyértelműen meghatároz egy lehetséges megoldás struktúrát, a lehetséges megoldás struktúrákat egy logikai konjunktív normálformával fogjuk jellemezni.

Legyen $\overline{O} = \{(\alpha_1,\beta_1),\ldots,(\alpha_l,\beta_l)\}$ és $J = \{1,\ldots,l\}$. Akkor az $(\overline{M},\overline{O})$ minden (m,o) folyamat részgráfjához hozzárendelhetünk egy $y_1,\ldots,y_l$ logikai vektort úgy, hogy minden $j \in J$-re $y_j = IGAZ \Longleftrightarrow (\alpha_j,\beta_j) \in o$. Könnyen belátható, hogy ez egy bijektív leképezés az $(\overline{M},\overline{O})$ folyamat $(\mathcal{A}4)$-et kielégítő részgráfjai és a hozzájuk rendelt logikai vektorok között. Egy $\mathbf{y}$ logikai vektornak megfelelő (m,o) folyamat részgráf a következőképpen határozható meg:

$$m = \bigcup_{j\in T(\mathbf{y})} (\alpha_j \cup \beta_j) \text{ és } o = \{(\alpha_j,\beta_j) : j \in T(\mathbf{y})\},$$

ahol $T(\mathbf{y}) = \{j : j \in J \text{ és } y_j = IGAZ\}$. Azonban tetszőleges $\mathbf{y}$ vektorhoz rendelt folyamat részgráf nem feltétlen lehetséges megoldás struktúra is. Az alábbiakban meghatározunk egy olyan Φ konjunktív normálformát, amelyet egy $\mathbf{y}$ logikai vektor akkor és csakis akkor elégít ki, ha a hozzárendelt folyamat részgráf lehetséges megoldás struktúra. Legyenek:

- $\Phi_0 = \bigwedge\limits_{X\in P} \bigvee\limits_{\substack{j\in J\\ X\in\beta_j}} y_j,$
- $\Phi_1 = \bigwedge\limits_{\substack{j\in J\\ X\in\alpha_j\setminus R}} (\neg y_j \vee \bigvee\limits_{\substack{h\in J\\ X\in\beta_h}} , y_h)$
- $\Phi_2 = \bigwedge\limits_{\substack{j\in J\\ P\cap\beta_j=\emptyset}} (\neg y_j \vee \bigvee\limits_{\substack{h\in J\\ \beta_j\cap\alpha_h\neq\emptyset}} y_h),$
- $\Phi = \Phi_0 \wedge \Phi_1 \wedge \Phi_2.$

3.2.1. Lemma. ([37]) *Egy* $\mathbf{y}$ *logikai* l*-vektor akkor és csakis akkor elégíti ki a* Φ *konjunktív normál formát, ha az* $\mathbf{y}$*-hoz rendelt folyamat gráf egy lehetséges megoldás struktúra.*

Bizonyítás Legyen (m,o) egy tetszőleges $(\overline{M},\overline{O})$-beli lehetséges megoldás struktúra és $\mathbf{y}$ a hozzárendelt logikai vektor. Mivel $m = mat(o)$, $(\mathcal{A}1)$ és $(\mathcal{A}2)$

alapján minden P-beli anyagot gyárt valamilyen o-beli műveleti egység, így $\Phi_0(\mathbf{y}) = IGAZ$. $(\mathcal{A}2)$-ből nyilván következik $\Phi_1(\mathbf{y}) = IGAZ$ is.

Bizonyítani szeretnénk Φ_2 teljesülését is. Legyen $j \in J$ úgy, hogy $\beta_j \cap P = \emptyset$. Akkor $\neg y_j \vee \bigvee_{\substack{h \in J \\ \beta_j \cap \alpha_h \neq \emptyset}} y_h$ tagja a Φ_2-nek. Ha $u_j \notin o$, akkor $y_j = HAMIS$ és az előbbi diszjunkció logikai értéke $IGAZ$. Meg kell még mutatnunk, hogy akkor is igaz, ha $u_j \in o$, azaz $y_j = IGAZ$. Ebben az esetben $(\mathcal{A}3)$ alapján létezik egy (m, o)-beli út u_j-ből P-be, ami miatt létezik olyan $u_h \in o$, melyre $\beta_j \cap \alpha_h \neq \emptyset$. Ez viszont azt jelenti, hogy $y_h = IGAZ$, amiből az következik, hogy az őt tartalmazó diszjunkció is igaz. Mivel a $j \in J$-t tetszőlegesen választottuk a $\beta_j \cap P = \emptyset$ feltétel mellett, így a Φ_2 konjunkció minden tagja $IGAZ$ lesz, amiből következik Φ_2 teljesülése.

A fentiekben megmutattuk tehát, hogy Φ_0, Φ_1 és Φ_2 mindegyike $IGAZ$, amiből következik Φ teljesülése.

A másik irány bizonyításához most tegyük fel, hogy $\mathbf{y}$ kielégíti Φ-t. Legyen (m, o) az $(\overline{M}, \overline{O})$ $\mathbf{y}$-hoz rendelt folyamat részgráfja. Igazolni szeretnénk, hogy (m, o) lehetséges megoldás struktúra, azaz kielégíti az $(\mathcal{A}1)$ - $(\mathcal{A}4)$ feltételeket.

$(\mathcal{A}4)$ az $\mathbf{y}$ definíciójából közvetlenül következik.

$\Phi_0(\mathbf{y}) = IGAZ$ miatt minden $X \in P$-re létezik olyan $u \in O$ műveleti egység, mely X-et közvetlenül termeli, így $(\mathcal{A}1)$ is teljesül.

$(\mathcal{A}2)$ bizonyításához legyen $X \in m \cap R$. Mivel $o \subseteq \overline{O}$ és az $(\overline{M}, \overline{O})$ maximális struktúrában a nyersanyagokat egyetlen $\overline{O}$-beli műveleti egység sem gyártja, így nem létezhet (Y, X) él az (m, o)-ban. Feltételezzük, hogy $X \in m$ és nem létezik (Y, X) él (m, o)-ban. Bizonyítani szeretnénk, hogy $X \in R$. Feltételezzük az ellenkezőjét, azt hogy $X \notin R$. Mivel $X \in m$, a Φ-hez rendelt (m, o) definíciója miatt léteznie kell $u_i = (\alpha_i, \beta_i) \in o$-nak úgy, hogy $X \in \alpha_i$, ami azt jelenti, hogy $y_i = IGAZ$. Másfelöl az $(\overline{M}, \overline{O})$ maximális struktúra is a 2.1.1. következmény alapján egy lehetséges megoldás struktúra, melyben feltételezésünk szerint X nem nyersanyag, így létezniük kell olyan $u_h \in \overline{O}$ műveleti egységeknek, melyekre $X \in \beta_h$. Mivel a Φ_1 a maximális struktúra minden, nem csak nyersanyag bemenettel rendelkező műveleti

egységére tartalmaz egy tagot, ezért Φ_1 tartalmazni fogja a $(\neg y_i \vee \bigvee_{\substack{h\in J\\ X\in\beta_h}} y_h)$ tagot is. Mivel feltételezésünk szerint $\Phi = IGAZ$, ezért $\Phi_1 = IGAZ$, de akkor az $(\neg y_i \vee \bigvee_{\substack{h\in J\\ X\in\beta_h}} y_h)$ tagnak is $IGAZ$-nak kell lennie. Ugyanakkor a fentiekben azt kaptuk, hogy $y_i = IGAZ$. Ez azt jelenti, hogy $\bigvee_{\substack{h\in J\\ X\in\beta_h}} y_h$ is $IGAZ$, amiből az következik, hogy létezik olyan h_0, melyre $u_{h_0} \in o$ és $X \in \beta_{h_0}$. Ebből viszont az adódik, hogy létezik (Y, X) él (m, o)-ban, ami ellentmond a feltételezésünknek. Következésképpen $X \in R$, és $(\mathcal{A}2)$ teljesül.

$(\mathcal{A}3)$ bizonyításához vegyünk egy tetszőleges $u_i \in o$ műveleti egységet. Ha $\beta_i \cap P \neq \emptyset$, akkor nyilván létezik út u_i-ből P-be. Egyébként, felhasználva azt a tényt, hogy a maximális struktúra egy lehetséges megoldás struktúra, a maximális struktúrában létezik út u_i-ből P-be, így létezik olyan $u_h \in \overline{O}$ műveleti egység, melyre $\beta_i \cap \alpha_h \neq \emptyset$. Mivel Φ_2 a maximális struktúra minden célterméket nem gyártó műveleti egységére tartalmaz egy tagot, ezért tartalmazni fogja a $\neg y_i \vee \bigvee_{\substack{h\in J\\ \beta_j\cap\alpha_h\neq\emptyset}} y_h$ tagot is. Mivel $u_i \in o$ miatt $y_i = IGAZ$, ezért az előbbiekhez hasonló módon létezik olyan h_0, hogy $u_{h_0} \in o$ és $\beta_i \cap \alpha_{h_0} \neq \emptyset$. Most u_{h_0}-al indulva ugyanezt a gondolatmenetet megismételve, véges számú ismétlő lépés után kapunk egy u_i-ből P-be vezető (m, o)-beli utat, ami azt mutatja, hogy $(\mathcal{A}3)$ igaz. □

A 3.2.1. lemma alapján felírhatjuk a PNS probléma egy ekvivalens formáját:

$$\min\left\{\sum_{j\in T(\mathbf{y})} w_j \ : \ \mathbf{y} \text{ kielégíti } \Phi\text{-t}\right\}, \tag{3.2}$$

ahol $w_j = w(u_j),\ j = 1, \ldots, l$.

A ([24]) alapján bevezetve a $z_j^+, z_j^- \in \{0, 1\},\ j = 1, \ldots, l$ változókat úgy, hogy $z_+^j = 1$ akkor és csakis akkor ha $y_j = IGAZ$, továbbá $z_j^- = 1 - z_j^+$, a fenti feladatot az alábbi formában is felírhatjuk:

$$\sum_{\substack{j\in J\\ X\in\beta_j}} z_j^+ \geq 1, \text{ minden } X \in P\text{-re},$$

$$z_j^- + \sum_{\substack{h \in J \\ X \in \beta_h}} z_h^+ \geq 1, \quad \text{minden } j \in J,\ X \in \alpha_j \setminus R\text{-re},$$

$$z_j^- + \sum_{\substack{h \in J \\ \beta_j \cap \alpha_h \neq \emptyset}} z_h^+ \geq 1, \text{ minden } j \in J,\ P \cap \beta_j = \emptyset\text{-re},$$

$$z_j^+ + z_j^- = 1, \text{ minden } j \in J\text{-re},$$

$$z_j^+, z_j^- \in \{0,1\}, \text{ minden } j \in J\text{-re},$$

$$\min \sum_{j \in J} w_j \cdot z_j^+$$

A fenti, $2l$ változót tartalmazó feltételrendszer minden változója bináris. A feltételrendszer mátrixa szintén bináris, minden feltételben mindegyik változó együtthatója 0 vagy 1. A jobboldali vektor minden komponense 1. Vannak egyenlőségek is, illetve ≥ 1 feltételek. Egy ilyen feltételrendszerű és célfüggvényű probléma nem halmazlefedési feladat az egyenlőségek miatt, de nem is halmazosztályozási feladat, mert nem minden feltétel egyenlőség. A pl. [36]-ban bemutatott bizonyítás alapján könnyen belátható, hogy tetszőleges $L > \sum_{j \in J} w_j$-re a fenti feladat optimális megoldásai azonosak az alábbi halmazlefedési probléma optimális megoldásaival:

$$\sum_{\substack{j \in J \\ X \in \beta_j}} z_j^+ \geq 1, \text{ minden } X \in P\text{-re},$$

$$z_j^- + \sum_{\substack{h \in J \\ X \in \beta_h}} z_h^+ \geq 1, \quad \text{minden } j \in J,\ X \in \alpha_j \setminus R\text{-re},$$

$$z_j^- + \sum_{\substack{h \in J \\ \beta_j \cap \alpha_h \neq \emptyset}} z_h^+ \geq 1, \text{ minden } j \in J,\ P \cap \beta_j = \emptyset\text{-re},$$

$$z_j^+ + z_j^- \geq 1, \text{ minden } j \in J\text{-re},$$

$$z_j^+, z_j^- \in \{0,1\}, \text{ minden } j \in J\text{-re},$$

$$\min \sum_{j \in J} [(w_j + L) \cdot z_j^+ + L \cdot z_j^-]$$

Összegezve az eddigieket, azt kaptuk, hogy a PNS probléma visszavezethető egy halmazlefedési feladatra, ugyanakkor a 3.1.1. lemma értelmében a halmazlefedési feladat ekvivalens egy speciális PNS problémaosztállyal. Ebből az következik, hogy:

3.2.1. Tétel. [37] *A PNS probléma ekvivalens a halmazlefedési problémával.*

A halmazlefedési probléma NP–teljessége (ld. [2, 41]) alapján ebből is adódik, hogy:

3.2.2. Tétel. [4], [37] *A PNS probléma NP–teljes.*

3.2.1. Megjegyzés. A [4] cikkben leírt visszavezetés, a [24, 37] cikkekben publikált, itt felidézett ekvivalencia bizonyítás a súlyozott PNS problémák közül az ún. Minsum problémát az NP–teljes problémák osztályába helyezi. A már egy ideje, különböző termelési és egyéb gyakorlati kérdések által motivált, és sokat vizsgált probléma elhelyezése a klasszikus, híres problémák közé fontos eredmény. Ezek után ugyanolyan joggal vethetők fel érdekes kérdések a PNS probléma osztállyal kapcsolatban (speciális struktúrák, egyedi célfüggvények), mint bármely más univerzálisan nehéz problémáról. A fenti ekvivalencia bizonyítás logikai formulákat használ. Ennek következtében talán lehetséges lett volna igazolni, hogy az általános PNS problémából adódó CNF van annyira általános, hogy kielégíthetősége még mindig NP–teljes. Ahhoz azonban, hogy a halmazlefedési problémával való ekvivalencia kiderüljön, a kulcs ötlet az, hogy ha a változókra a $z_+^j = 1$ pontosan akkor teljesül, ha az u_j-t beválasztjuk a lehetséges megoldásba, akkor a lehetséges megoldásra vonatkozó axiómák éppen a megadott feltételrendszert eredményezik. A feltételrendszer éppen olyanná volt alakítható, mint amilyennek egy halmazlefedési feladatnak lennie kell. A logikai formulára való visszavezetést, a két modell ekvivalenciáját a történeti hűség kedvéért mutattuk be.

A 3.2.2. tétel alapján feltételezhető, hogy a PNS probléma hatékony, polinomiális idejű megoldása nem várható. Ez indokolja azt, hogy akár exponenciális idejű megoldó algoritmusokat is elfogadhatónak tekintsünk, és alkalmazzuk az ilyen esetekben szokásos korlátozás és szétválasztás jellegű módszereket.

4. fejezet

Döntési leképezések

Fontos szerepet játszik a PNS-problémának a korlátozás és szétválasztás módszerével történő, különböző ([34]) megoldásaiban a ([20, 21])-ban bevezetett döntési leképezés fogalma.

Legyen O a műveleti egységek egy halmaza. Definiáljuk a $\Delta : M \setminus R \longrightarrow \wp(O)$, függvényt a következő módon. Minden $X \in M \setminus R$-re legyen $\Delta(X) = \{(\alpha, \beta) : (\alpha, \beta) \in O \ \& \ X \in \beta\}$. Szemléletesen a Δ minden nem nyersanyaghoz hozzárendeli azokat a műveleti egységeket, melyek azt az anyagot gyárthatják. Mi viszont általános esetben a műveleti egységeknek csak egy olyan részhalmazát szeretnénk használni, mely a legkisebb költséggel képes a céltermékek legyártására. Mivel általában egy anyagot több műveleti egység is legyárthat, ezért egy megoldás felépítése során minden anyagra vonatkozóan döntenünk kell arról, hogy azt az anyagot mely műveleti egységekkel kívánjuk legyártani. Ezen döntések leírására használjuk a döntési leképezéseket. Legyen $m \subseteq M \setminus R$ és $\delta(X) \subseteq \Delta(X)$, minden $X \in m$-re. A $\delta[m] = \{(X, \delta(X)) : X \in m\}$ leképezést ***reguláris döntési leképezésnek***, vagy egyszerűen csak ***döntési leképezésnek*** nevezzük.

4.1. Konzisztens döntési leképezés

Egy döntési leképezés ***konzisztens***, ha $\delta(X) \cap \Delta(Y) \subseteq \delta(Y)$ bármely $X, Y \in m$-re. Ez azt fejezi ki, hogy ha egy műveleti egység több anyagot is gyárt,

akkor vagy azt feltételezzük, hogy nem működik, azaz nem gyárt semmit és akkor nem rendeljük hozzá egyetlen anyaghoz sem, vagy azt feltételezzük, hogy működik, de akkor mindegyik kimeneti anyagát gyártja és így mindhez hozzá kell rendelni. Az $\mathbf{M}$ strukturális modell konzisztens döntési leképezéseinek halmazát $\Omega_{\mathbf{M}}$-el fogjuk jelölni.

Most tegyük fel, hogy egy $\delta[m]$ döntési leképezés által meghatároztuk a műveleti egységek egy részhalmazát abból a célból, hogy az m-beli anyagokat közvetlenül gyártsa. Ha veszünk egy további $Y \in M \setminus (m \cup R)$ anyagot, és ennek közvetlen gyártására konzisztens módon hozzárendeljük az $u'_1, \ldots, u'_r$ $\Delta(Y)$-beli bizonyos műveleti egységeket, akkor egy bővebb döntési leképezést kapunk. Az ennek megfelelő

$$\delta'[m \cup \{Y\}] = \delta[m] \cup \{(Y, \{u'_1, \ldots, u'_r\})\}$$

döntési leképezésre azt mondjuk, hogy a $\delta[m]$ egy ***reguláris kiterjesztése***, vagy egyszerűen csak a $\delta[m]$ ***kiterjesztése***.

A kiterjesztés függvény $\Omega_{\mathbf{M}}$-en reflexív, antiszimmetrikus és tranzitív, tehát részben rendezés relációt határoz meg. Jelöljük a részben rendezett halmazt $(\Omega_{\mathbf{M}}, \leq)$-el. Nyilvánvalóan $\delta[\emptyset]$ az $(\Omega_{\mathbf{M}}, \leq)$ halmaz legkisebb eleme. A fentiek értelmében a kiterjesztés relációt általánosíthatjuk úgy, hogy azt mondjuk, hogy $\delta_2[m_2]$ ***(reguláris) kiterjesztése*** $\delta_1[m_1]$-nek és ezt ugyancsak $\delta_1[m_1] \leq \delta_2[m_2]$ -vel jelöljük, ha $m_1 \subseteq m_2$, $\delta_1[m_1]$ és $\delta_2[m_2]$ konzisztens döntési leképezések, valamint $\delta_1(X) = \delta_2(X)$ minden $X \in m_1$-re. Az $(\Omega_{\mathbf{M}}, \leq)$ halmaz maximális elemeit $\Omega_{\mathbf{M}}^{\max}$-val fogjuk jelölni.

Könnyen beláthatjuk, hogy teljesül az alábbi tulajdonság:

4.1.1. Lemma. *Egy $\delta[m] \in \Omega_{\mathbf{M}}$ döntési leképezés az $(\Omega_{\mathbf{M}}, \leq)$ részben rendezett halmaz maximális eleme akkor és csakis akkor, ha $m = M \setminus R$.*

Érdemes definiálni a műveleti egységek azon halmazát, amely egy konzisztens döntési leképezés által van meghatározva. Nevezetesen, legyen

$$op(\delta[m]) = \cup \{\delta(X) : X \in m\}.$$

Bizonyos esetekben egy $\delta[m] \in \Omega_{\mathbf{M}} \setminus \Omega_{\mathbf{M}}^{\max}$ döntési leképezésből maximális elemet képezhetünk a következő módon. Ha

$$W = (mat^{in}(op(\delta[m])) \cup P) \setminus (R \cup m) = \emptyset,$$

akkor legyen

$$\delta'(X) = \{(\alpha, \beta) : (\alpha, \beta) \in op(\delta[m]) \text{ s } X \in \beta\}$$

minden $X \in M \setminus R$-re. Akkor könnyen belátható, hogy $\delta' \in \Omega_{\mathbf{M}}^{\max}$. A δ' döntési leképezést a $\delta[m]$ ***(reguláris) lezárásának*** fogjuk nevezni.

Mivel minden $\delta[m] \in \Omega_{\mathbf{M}}^{\max}$-re $m = M \setminus R$, ezért ismert $\mathbf{M} = (P, R, O)$ strukturális modell esetén adott maximális döntési leképezést egyszerűen δ-val fogunk jelölni.

A maximális konzisztens döntési leképezések és a lehetséges megoldás struktúrák közötti összefüggések tanulmányozására minden $(m, o) \in S(\mathbf{M})$ lehetséges megoldás struktúrához hozzárendelünk egy maximális konzisztens döntési leképezést a következőképpen.

4.1.1. Definíció. *Legyen $\rho : S(\mathbf{M}) \longrightarrow \Omega_{\mathbf{M}}^{\max}$ függvény, melyre $\rho(m, o) = \delta$ úgy, hogy*

$$\delta(X) = \{u : u = (\alpha, \beta) \in o \,\&\, X \in \beta\}, \text{ ha } X \in m \setminus R, \text{ és}$$

$$\delta(X) = \emptyset, \text{ ha } X \notin M \setminus (R \cup m).$$

4.1.2. Lemma. ([33]) *ρ egy injektív leképezés $S(\mathbf{M})$-ről $\Omega_{\mathbf{M}}^{\max}$-ba, továbbá*

$$\rho^{-1}(\delta) = (mat(op(\delta)), op(\delta))$$

igaz minden olyan δ-ra, mely egy $S(\mathbf{M})$-beli elem ρ általi leképezése.

Bizonyítás Először megmutatjuk, hogy $(m, o) \in S(\mathbf{M})$-ből következik $\rho((m, o)) \in \Omega_{\mathbf{M}}^{\max}$. Mivel a $\rho((m, o)) = \delta[M \setminus R]$ az $M \setminus R$ halmazon definiált, továbbá egy $\delta[m]$ döntési leképezés akkor és csakis akkor maximális, ha $m = M \setminus R$, ezért azt kell belátnunk, hogy δ konzisztens. Ebből a célból legyenek $X, Y \in M \setminus R$ tetszőleges anyagok. Igazoljuk, hogy $\delta(X) \cap \Delta(Y) \subseteq \delta(Y)$. Ha $\delta(X) \cap \Delta(Y) = \emptyset$, akkor a konzisztenciához szükséges tartalmazás nyilvánvalóan teljesül. Most tegyük fel, hogy $\delta(X) \cap \Delta(Y) \neq \emptyset$ és $(\alpha, \beta) \in \delta(X) \cap \Delta(Y)$. Akkor δ definíciója alapján $X \in m \setminus R$ és $(\alpha, \beta) \in o$.

Másfelöl $(\alpha, \beta) \in \Delta(Y)$-ből következik, hogy $Y \in \beta$. Mivel (m, o) egy megoldás struktúra, ezért a 2.1.2. Lemma alapján $m = mat(o) \supseteq \beta \ni Y$, és így $Y \in m$. Az $Y \in M \setminus R$ feltételezés azt vonja maga után, hogy $Y \notin R$, és ezért $Y \in m \setminus R$. Azt kapjuk tehát, hogy $(\alpha, \beta) \in \delta(Y)$, amiből következik az előírt tartalmazás.

Most legyen (m, o) és (m', o') két különböző megoldás struktúra. Megmutatjuk, hogy a $\rho((m, o)) = \delta$ és $\rho((m', o')) = \delta'$, egymástól különböző döntési leképezések. A 2.1.2. Lemma és 2.1.2. Következmény alapján az (m, o) és (m', o') egyértelműen meghatározottak az o illetve o' halmazok által, ezért $o \neq o'$. Akkor az általánosság megszorítása nélkül feltételezhetjük, hogy létezik olyan $(\alpha, \beta) \in O$, melyre $(\alpha, \beta) \in o$, de $(\alpha, \beta) \notin o'$. Mivel $(m, o) \in S(\mathbf{M})$, ezért a 2.1.2. Lemma alapján $m = mat(o)$ és így $\beta \subseteq m$. Az $(\mathcal{A}3)$ feltételből $|\beta| \geq 1$, vagyis létezik legalább egy olyan X anyag, melyre $X \in \beta$, és ezért $X \in m$. Mivel (M, O) egy megoldás struktúra és $X \in \beta$, továbbá $(\alpha, \beta) \in O$, ezért az $(\mathcal{A}2)$ feltételből $X \notin R$ és így $X \in m \setminus R$. Továbbá a δ definíciójából $(\alpha, \beta) \in \delta(X)$. Két esetet különböztethetünk meg.

Ha $X \notin m'$, akkor a δ' definíciója alapján $\delta'(X) = \emptyset$, következésképpen $\delta(X) \neq \delta'(X)$.

Ha $X \in m'$, akkor $(\alpha, \beta) \notin o'$-ből δ' definíciója alapján következik, hogy $(\alpha, \beta) \notin \delta'(X)$.

Ily módon igazoltuk, hogy $\delta \neq \delta'$.

Végül legyen $\delta = \rho((m, o))$ valamely $(m, o) \in S(\mathbf{M})$ megoldás struktúrára. A ρ definíciója alapján könnyen belátható, hogy $o = op(\delta)$. Akkor a 2.1.2. Lemmából azt kapjuk, hogy $(m, o) = (mat(op(\delta)), op(\delta))$, ami azt jelenti, hogy igazoltuk az állítást. □

4.1.1. Megjegyzés. Vegyük észre, hogy ρ nem ráképezés, és így nem is bijekció. Például a $\delta^*(X) = \emptyset, \forall X \in M \setminus R$ döntési leképezés eleme az $\Omega_{\mathbf{M}}^{\max}$ halmaznak, de nem létezik olyan $(m, o) \in S(\mathbf{M})$ lehetséges megoldás struktúra, melyre $\rho((m, o)) = \delta^*$ teljesülne.

Jelölje $S'(\mathbf{M})$ az $S(\mathbf{M})$ ρ melletti képét, vagyis legyen

$$S'(\mathbf{M}) = \{\rho((m, o)) : (m, o) \in S(\mathbf{M})\}.$$

Vegyük észre, hogy minden $(m, o) \in S(\mathbf{M})$-re ha $\rho((m, o)) = \delta[M \setminus R]$, akkor $o = op(\delta[M \setminus R])$. Ezen észrevételből és a 4.1.2. Lemmából azt kapjuk, hogy a PNS feladattal az alábbi is egyenértékű megfogalmazás:

$$\min \left\{ \sum_{u \in op(\delta)} w(u) : \delta \in S'(\mathbf{M}) \right\}. \tag{4.1}$$

Az egyszerűség kedvéért az $S'(\mathbf{M})$ elemeit **lehetséges megoldások**nak fogjuk nevezni.

A későbbiekben látni fogjuk, hogy a konzisztens döntési leképezések lehetővé teszik a lehetséges megoldás struktúrák hatékonyabb leírását és generálását.

4.2. Megoldó eljárások

Ebben a részben a (4.1) feladat megoldására fogunk kidolgozni alapvető algoritmusokat. A 3.2.2 Tétel alapján tudjuk, hogy nem várható hatékony polinomiális idejű megoldás. Ismertetünk egy olyan Branch and Bound jellegű eljárást, mely általános keretet nyújt majd az ilyen algoritmusok későbbiekben történő tárgyalására.

4.3. Branch and Bound kereteljárás PNS feladatokra

Legyen $\mathbf{M}$ egy PNS probléma strukturális modellje, és jelölje (M, O) a maximális struktúrát. Célunk az, hogy megoldjuk a (4.1) problémát. Mivel az $S'(\mathbf{M})$ nem üres, véges halmaz, ezért léteznie kell legalább egy optimális megoldásnak. A továbbiakban megmutatjuk, hogy hogyan alkalmazható az általános korlátozás és szétválasztás módszere a feladat megoldására.

A Branch and Bound algoritmus egyik fő összetevője a szétválasztási függvény, mely $S'(\mathbf{M})$ részhalmazait partíciókra osztja. Mivel itt $S'(\mathbf{M})$ csak implicit módon van definiálva, viszont $S'(\mathbf{M}) \subseteq \Omega_{\mathbf{M}}^{\max}$, amiből kifolyólag az

$\Omega_{\mathbf{M}}^{\max}$ partíciói az $S'(\mathbf{M})$-ben is partíciókat hoznak létre, ezért $S'(\mathbf{M})$ helyett az őt befoglaló $\Omega_{\mathbf{M}}^{\max}$ halmazt és annak részhalmazait fogjuk felosztani.

Definiáljuk az $\omega : \Omega_{\mathbf{M}} \setminus \Omega_{\mathbf{M}}^{\max} \longrightarrow \varphi'(\Omega_{\mathbf{M}}^{\max})$ függvényt a következőképpen. Minden $\delta[m] \in \Omega_{\mathbf{M}} \setminus \Omega_{\mathbf{M}}^{\max}$ -re legyen

$$\omega(\delta[m]) = \{\delta : \ \delta \in \Omega_{\mathbf{M}}^{\max} \text{ és } \delta[m] \leq \delta\}.$$

Tulajdonképpen $\omega(\delta[m])$ megadja a $\delta[m]$ maximális konzisztens kiterjesztéseit. A 4.1 részben láttuk, hogy minden nem maximális konzisztens döntési leképezésnek létezik maximális konzisztens döntési leképezésre való kiterjesztése. Az is nyilvánvaló, hogy $\omega(\delta[\emptyset]) = \Omega_{\mathbf{M}}^{\max}$. Az $\omega(\delta[m]) \cap S'(\mathbf{M})$ halmaz elemeit ***(reguláris) lehetséges megoldás kiterjesztések***nek fogjuk nevezni.

A [35]-ban igazolást nyert a következő állítás.

4.3.1. Lemma. *Ha* $\delta[m], \delta'[m'] \in \Omega_{\mathbf{M}} \setminus \Omega_{\mathbf{M}}^{\max}$, *és* $\exists X \in m \cap m' : \ \delta[X] \neq \delta'[X]$, *akkor* $\omega(\delta[m]) \cap \omega(\delta'[m']) = \emptyset$.

Legyen $\delta[m] \in \Omega_{\mathbf{M}} \setminus \Omega_{\mathbf{M}}^{\max}$. Akkor a 4.1.1. Lemma értelmében $m \subset M \setminus R$, ami azt jelenti, hogy létezik $X \in M \setminus (R \cup m)$ anyag. Mivel (M, O) maximális struktúra, ezért $\Delta(X) \neq \emptyset$, így $\Delta(X)$ minden Q_i, $i = 1, \ldots, 2^{|\Delta(X)|}$ részhalmazára definiálhatjuk a

$$\delta_i[m \cup \{X\}] = \delta[m] \cup \{(X, Q_i)\}$$

döntési leképezést.

Az általánosság megszorítása nélkül feltételezhetjük, hogy a $\delta_i[m \cup \{X\}]$, $i = 1, \ldots, 2^{|\Delta(X)|}$ halmaz konzisztens döntési leképezései a $\delta_t[m \cup \{X\}]$, $t = 1, \ldots, k$ döntési leképezések. Akkor a [35] alapján tudjuk, hogy

4.3.2. Lemma. *Az* $\omega(\delta_t[m \cup \{X\}])$, $t = 1, \ldots, k$ *halmazok nem feltétlen nem triviális partíciói az* $\omega(\delta[m])$ *halmaznak.*

Akkor a B&B algoritmus ϖ szétválasztási függvénye a következő lesz:

$$\varpi(\delta[m]) = \{\omega(\delta_t[m \cup \{X\}]), \ t = 1, \ldots, k\}.$$

A Branch and Bound algoritmus másik lényeges összetevője a korlátozó függvény. Jelen esetben ez egy $g : \Omega_{\mathbf{M}} \longrightarrow \mathbf{R}$ függvény, mely alsó korlátokat határoz meg a $w(\delta')$, $\delta' \in S'(\mathbf{M}) \cap \omega(\delta[m])$, $\delta[m] \in \Omega_{\mathbf{M}} \setminus \Omega_{\mathbf{M}}^{\max}$ értékekre, $g(\delta) = w(\delta)$ ha $\delta[m] \in S'(\mathbf{M})$, és $g(\delta) = \infty$ ha $\delta[m] \in \Omega_{\mathbf{M}}^{\max} \setminus S'(\mathbf{M})$.

A korlátozás és szétválasztás módszere tulajdonképpen az összes lehetséges megoldást tartalmazó leszámlálási fa olyan intelligens bejárása, mely a szétválasztó és korlátozó függvényeinek köszönhetően a fának csak egy részét generálja és járja be azért, hogy minél hamarabb eljusson az optimális megoldáshoz. Ez úgy lehetséges, hogy azokat a csúcspontokat nem fogja tovább osztani, melyekről a korlátozó függvény segítségével biztosan meg tudja állapítani, hogy nem tartalmazhatnak optimális megoldást. Az ilyen csúcspontokat ***felderített***, vagy ***lezárt*** csúcspontoknak szoktuk nevezni, míg a többi csúcspontok alkotják az ***élő*** csúcspontokat. Ezek után megadhatjuk a (4.1) problémát megoldó B&B algoritmust.

Branch and Bound Algoritmus ([35])

Inicializálás

- Legyen $m = \emptyset$, $L = \{\omega(\delta[\emptyset])\}$, $z = \infty$, $s = \emptyset$, $r = 0$.
  Számoljuk ki $g(\delta[\emptyset])$-t.

Iteráció (r. iteráció)

1. *Befejezés tesztelés*

   Ha $L = \emptyset$, akkor VÉGE: z tartalmazza az optimumot, s pedig az optimális megoldást.
   Egyébként, ha $r > 0$, akkor térjünk a 2. lépésre,
   ha pedig $r = 0$, akkor folytassuk a 3. lépéssel.

2. *Levélkiválasztás*

   Válasszunk egy olyan $\omega(\delta[m])$ elemet az L-ből, melyre $\frac{g(\delta[m])}{|m|}$ minimális.
   Ha több ilyen van, akkor válasszunk egyet véletlenszerűen közülük.

3. *Szétválasztás*

   Alkossuk meg $\omega(\delta[m])$-nek a $\varpi(\delta[m]) = \{\omega(\delta_i[m_i]),\ i = 1, \ldots, k\}$ partícióit.

4. *Korlátszámítás*

   Minden $i = 1, \ldots, k$-ra számoljuk ki a $g(\delta_i[m_i])$ korlátokat. Továbbá, ha $m = M \setminus R$ és $g(\delta_i) < z$, akkor legyen $z = g(\delta_i)$ és $s = \{\delta_i\}$.

5. *Felderítés*

   Aktualizáljuk L értékét:

   $$L = \{\omega(\delta'[m']) :\ \omega(\delta'[m']) \in (L \setminus \{\omega(\delta[m])\}) \cup \varpi(\delta[m]),\ g(\delta'[m']) < z\}$$

   Legyen $r = r + 1$, és térjünk a következő iterációra (1. lépés).

4.3.1. Megjegyzés. A leírt algoritmus nem teljes, hanem egy séma, mely a különböző választási lehetőségek pontosításával implementálható. Például a 3. lépésben nem határoztuk meg, hogy mely anyaggal fogjuk kiterjeszteni az m anyaghalmazt a partíciók képzése céljából. Erre egy lehetőség lehet, hogy az $M \setminus (m \cup R)$ halmaz legkisebb indexű elemét választjuk. Hasonlóképpen, a g korlátozó függvény pontos képletét sem adtuk meg. Triviális korlátként megadhatnánk például a $\delta[m]$ által rögzített műveleti egységek súlyainak összegét. Ezek különböző implementációja különböző B&B algoritmusokhoz vezet. További ilyen lehetőségeket tartalmaz a [35] munka.

5. fejezet

A döntési leképezések száma

A fejezet a [5], [6] és [7] cikkek eredményeit tartalmazza. Ezek olyan közös publikációk, melyek eredményeit szerzőtársaimmal oszthatatlanoknak tekintjük.

A konzisztencia szükséges ahhoz, hogy eljussunk a lehetséges megoldásokig. Azonban több is igaz: minden konzisztens döntési leképezés azonosítható a fentiek szerint hozzárendelt műveleti egységek részhalmazával, hiszen, ha $m \subseteq M \setminus R$ adott, akkor azon műveleti egységek közül választhatunk, amelyek kimenetében szerepel legalább egy anyag m-ből. Viszont ha egy műveleti egységet valamelyik anyaghoz hozzárendeltük, akkor éppen a konzisztencia miatt ezt már minden kimeneti anyagához hozzá kell rendelnünk. Ez az egyszerű észrevétel tehát azt jelenti, hogy az eddigieknél jobban kezelhetjük a konzisztens döntési leképezéseket és megszámolhatjuk őket.

5.1. Az általános eset

5.1.1. Tétel. ([5]) *Minden $\emptyset \neq m \subseteq M \setminus R$-re, az m-en definiálható konzisztens döntési leképezések száma* $2^{|\bigcup_{X \in m} \Delta(X)|}$.

Bizonyítás

Jelöljük $\tau(m)$-el az m anyaghalmaz felett definiálható konzisztens

döntési leképezések számát. Eljárásunkban $|m|$ szerinti indukciót fogunk alkalmazni.

Ha $|m| = 1$, akkor $X \to Q$ egy konzisztens döntési leképezés bármely $Q \subseteq \Delta(X)$-re, ahol X az m egyetlen elemét jelöli. A konzisztens döntési leképezések száma ebben az esetben $2^{|\Delta(X)|}$.

Most legyen $1 \leq i < |M \setminus R|$ egy tetszőleges egész szám, és feltételezzük, hogy az állítás igaz minden olyan $m \subseteq M \setminus R$-re, melyre $|m| = i$. Vegyünk egy tetszőleges $i+1$ elemű $m' \subseteq M \setminus R$ halmazt. Az általánosság megszorítása nélkül feltételezhetjük, hogy $m' = \{X_1, \ldots, X_i, X_{i+1}\}$. Legyen $W = \Delta(X_{i+1}) \setminus (\cup\{\Delta(X_t) : t = 1, \ldots, i\})$. W alapján két esetet különböztetünk meg.

1. ESET. $W = \emptyset$. Akkor $\bigcup_{X \in m'} \Delta(X) = \bigcup_{X \in m} \Delta(X)$.

Igazolnunk kell hát, hogy $\tau(m') = \tau(m)$. A konzisztencia definíciójából következik, hogy minden $\delta[m']$ konzisztens döntési leképezésnek az $\{X_1, \ldots, X_i\}$ halmazra való szűkítése is konzisztens döntési leképezés. Másfelől, ha azonos halmazon definiált két konzisztens döntési leképezés különböző, akkor a kiterjesztéseik is különbözőek lesznek. Elegendő tehát azt igazolni, hogy bármely $\delta[\{X_1, \ldots, X_i\}]$ döntési leképezésnek egyetlen kiterjesztése van az $\{X_1, \ldots, X_i, X_{i+1}\}$ halmazra.

Először megkonstruáljuk a $\delta[\{X_1, \ldots, X_i\}]$ egy kiterjesztését az $\{X_1, \ldots, X_i, X_{i+1}\}$ halmazra. Legyen

$$\delta'(X_{i+1}) = \{(\alpha, \beta) : (\alpha, \beta) \in \Delta(X_{i+1}) \;\&\; \exists j \in \{1, \ldots, i\} : \; (\alpha, \beta) \in \delta(X_j)\},$$

és

$$\delta'(X_t) = \delta(X_t), \;\; \forall t \in \{1, \ldots, i\}.$$

Először igazolnunk kell a $\delta'[\{X_1, \ldots, X_i, X_{i+1}\}]$ konzisztenciáját, nevezetesen azt, hogy

$$(1) \qquad \delta'(X_t) \cap \Delta(X_{i+1}) \subseteq \delta'(X_{i+1}),$$

és

$$(2) \qquad \delta'(X_{i+1}) \cap \Delta(X_t) \subseteq \delta'(X_t)$$

teljesülnek bármely $X_t \in \{X_1, \ldots, X_i\}$ esetén.

Az (1) érvényessége a δ' definíciójából következik. A (2) igazolására vegyünk egy $(\alpha, \beta) \in \delta'(X_{i+1}) \cap \Delta(X_t)$ tetszőleges műveleti egységet valamely $t \in \{1, \ldots, i\}$-re. Mivel $(\alpha, \beta) \in \delta'(X_{i+1})$, ezért $\exists j \in \{1, \ldots, i\}$: $(\alpha, \beta) \in \delta(X_j) \cap \Delta(X_{i+1})$. Akkor $(\alpha, \beta) \in \delta(X_j) \cap \Delta(X_t)$. Másfelöl, mivel $j, t \in \{1, \ldots, i\}$ ezért δ konzisztenciájából következik, hogy $\delta(X_j) \cap \Delta(X_t) \subseteq \delta(X_t) = \delta'(X_t)$. Következésképpen $(\alpha, \beta) \in \delta'(X_t)$, ami azt jelenti, hogy a (2) feltétel is teljesül.

Most legyen $\delta^*[\{X_1, \ldots, X_i, X_{i+1}\}]$ a $\delta[\{X_1, \ldots, X_i\}]$ egy kiterjesztése. Meg fogjuk mutatni, hogy $\delta'(X_t) = \delta^*(X_t)$, $\forall t \in \{1, \ldots, i+1\}$. Ha $1 \leq t \leq i$, akkor az egyenlőség nyilvánvalóan teljesül. Azt kell igazolnunk tehát, hogy $\delta'(X_{i+1}) \subseteq \delta^*(X_{i+1})$ és $\delta'(X_{i+1}) \supseteq \delta^*(X_{i+1})$. Ennek érdekében legyen $(\alpha, \beta) \in \delta'(X_{i+1})$ egy tetszőleges műveleti egység. A δ' definíciója alapján $(\alpha, \beta) \in \delta(X_j) \cap \Delta(X_{i+1})$ valamely $X_j \in \{X_1, \ldots, X_i\}$-re. De $\delta(X_j) = \delta^*(X_j)$ és δ^* konzisztens döntési leképezés, következésképpen $(\alpha, \beta) \in \delta^*(X_j) \cap \Delta(X_{i+1}) \subseteq \delta^*(X_{i+1})$. Fordítva, legyen $(\alpha, \beta) \in \delta^*(X_{i+1})$. Mivel $W = \emptyset$, ezért $\exists\ j \in \{1, \ldots, i\}$: $(\alpha, \beta) \in \Delta(X_j)$, és ily módon $(\alpha, \beta) \in \delta^*(X_{i+1}) \cap \Delta(X_j)$. De δ^* konzisztenciája miatt $\delta^*(X_{i+1}) \cap \Delta(X_j) \subseteq \delta^*(X_j) = \delta(X_j)$, következésképpen $(\alpha, \beta) \in \Delta(X_{i+1}) \cap \delta(X_j)$, de akkor δ' definíciója miatt $(\alpha, \beta) \in \delta'(X_{i+1})$.

2. ESET. $W \neq \emptyset$. Az 1. ESET.-ben tett észrevételek alapján elegendő azt megmutatni, hogy a $\delta[\{X_1, \ldots, X_i\}]$ konzisztens döntési leképezésnek $2^{|W|}$ kiterjesztése van a $\{X_1, \ldots, X_i, X_{i+1}\}$ halmazra. Ennek érdekében legyen

$$T = \{(\alpha, \beta) : (\alpha, \beta) \in \Delta(X_{i+1}) \,\&\, \exists t \in \{1, \ldots, i\} :\ (\alpha, \beta) \in \delta(X_t)\}.$$

T és W definíciójából nyilvánvaló, hogy $T \cap W = \emptyset$. Meg fogjuk mutatni, hogy a

$$\delta'(X) = \begin{cases} \delta(X), & \text{ha } X \in \{X_1, \ldots, X_i\} \\ T \cup Q, & \text{ha } X = X_{i+1} \end{cases}$$

döntési leképezés minden $Q \subseteq W$-re konzisztens. A $\delta[\{X_1, \ldots, X_i\}]$ konzisztenciája miatt elegendő igazolnunk a

$$\delta'(X_j) \cap \Delta(X_{i+1}) \subseteq \delta'(X_{i+1}) \tag{3}$$

és

$$\delta'(X_{i+1}) \cap \Delta(X_j) \subseteq \delta'(X_j) \tag{4}$$

tartalmazásokat minden $j = 1, \ldots, i$-re.

Ezért legyen $j \in \{1, \ldots, i\}$ egy tetszőleges index és $(\alpha, \beta) \in \delta'(X_j) \cap \Delta(X_{i+1})$. Akkor $(\alpha, \beta) \in T$, és így $(\alpha, \beta) \in \delta'(X_{i+1})$, amiből következik (3). Most legyen $(\alpha, \beta) \in \delta'(X_{i+1}) \cap \Delta(X_j)$. Akkor $(\alpha, \beta) \in (T \cup Q) \cap \Delta(X_j) = T \cap \Delta(X_j)$. Az $(\alpha, \beta) \in T$ tartalmazásból következik, hogy $(\alpha, \beta) \in \delta(X_t)$ valamely $t \in \{1, \ldots, i\}$-re, következésképpen $(\alpha, \beta) \in \delta(X_t) \cap \Delta(X_j)$. Mivel δ konzisztens, ezért $\delta(X_t) \cap \Delta(X_j) \subseteq \delta(X_j) = \delta'(X_j)$, ami a (4) teljesülését jelenti.

Mivel W lehetséges Q részhalmazainak száma $2^{|W|}$, ezért a fenti módszerrel a $\delta[\{X_1, \ldots, X_i\}]$ döntési leképezésnek $2^{|W|}$ különböző kiterjesztését kapjuk. Meg kell még mutatnunk, hogy a fenti döntési leképezésnek nincsenek további konzisztens kiterjesztései $\{X_1, \ldots, X_i, X_{i+1}\}$-re.

Legyen ehhez $\delta^*[\{X_1, \ldots, X_i, X_{i+1}\}]$ egy tetszőleges kiterjesztése δ-nak, és $(\alpha, \beta) \in T$. Akkor $(\alpha, \beta) \in \delta(X_t) \cap \Delta(X_{i+1}) = \delta^*(X_t) \cap \Delta(X_{i+1})$ valamely $t \in \{1, \ldots, i\}$-re. Mivel δ^* konzisztens, ezért $\delta^*(X_t) \cap \Delta(X_{i+1}) \subseteq \delta^*(X_{i+1})$, és így $(\alpha, \beta) \in \delta^*(X_{i+1})$. Következésképpen $T \subseteq \delta^*(X_{i+1})$. Most legyen $(\alpha, \beta) \in \delta^*(X_{i+1}) \setminus T$. Ha $(\alpha, \beta) \notin W$, akkor $(\alpha, \beta) \in \Delta(X_t)$ valamely $t \in \{1, \ldots, i\}$-re, és akkor δ^* konzisztenciája miatt $(\alpha, \beta) \in \delta^*(X_t) = \delta(X_t)$, ami azt jelenti, hogy $(\alpha, \beta) \in T$, de ez ellentmondás. Tehát $(\alpha, \beta) \in W$, így $\delta^*(X_{i+1}) \subseteq T \cup W$. Ez azt jelenti, hogy δ^*-nak meg kell egyezni δ valamely előzőekben már felsorolt kiterjesztésével.

Az indukciós feltevést használva tehát azt kapjuk, hogy

$$\tau(\{X_1, \ldots, X_{i+1}\}) = 2^{|\cup\{\Delta(X_t):t=1,\ldots,i\}|}2^{|W|} = 2^{|\cup\{\Delta(X_t):t=1,\ldots,i+1\}|},$$

amivel igazoltuk az állítást. □

Megjegyezzük, hogy m-en összesen $2^{\sum_{X\in m}|\Delta(X)|}$ döntési leképezés definiálható. Tételünkből következik, hogy $m = M \setminus R$-re, vagyis a maximális konzisztens döntési leképezésekre $\tau(\Omega_{\mathbf{M}}^{\max}) = 2^{|O|}$ adódik. Ez azt mutatja, hogy szoros kapcsolat áll fenn az O részhalmazai és a maximális konzisztens döntési leképezések között. Könnyen belátható, hogy $\gamma(\delta) = op(\delta)$ egy bijektív leképezés $\Omega_{\mathbf{M}}^{\max}$ és $\varphi(O)$ között.

A 4.1.1. Definícióban meghatároztunk egy ρ függvényt a lehetséges megoldás struktúrák és a maximális konzisztens döntési leképezések között. Világos, hogy ρ az $S(\mathbf{M})$-et kölcsönösen egyértelműen beleképezi $\Omega_{\mathbf{M}}^{\max}$-be. Ez viszont azt is jelenti, hogy $2^{|O|}$ egy triviális felső korlát $S(\mathbf{M})$-re. Természetesen ez nagyon durva becslés. Azonban ha akár már csak $(A2)$-t figyelembe vesszük, sokkal jobb becslés adható $|S(\mathbf{M})|$-re. A [5] munka alapján ezt fogjuk megvizsgálni a továbbiakban.

Legyen $(m, o) \in S(\mathbf{M})$ egy tetszőleges lehetséges megoldás struktúra és $\rho(m, o) = \delta$. Ha $X \in mat^{in}(op(\delta))$, akkor létezik $u = (\alpha, \beta) \in op(\delta)$ úgy, hogy $X \in \alpha$. A δ definíciója szerint $u \in o$, és így $X \in m$. $(\mathcal{A}2)$ alapján $X \in mat^{out}(op(\delta)) \cup R$, tehát felírhatjuk, hogy:

$$(\mathcal{A}'2) \qquad mat^{in}(op(\delta)) \subseteq mat^{out}(op(\delta)) \cup R.$$

Az $(\mathcal{A}'2)$-nek megfelelő maximális konzisztens döntési leképezések száma nyilván nem kevesebb, mint a lehetséges megoldás struktúrák száma, így felülről becsülünk, ha az előbbit meghatározzuk. Ennek érdekében legyen (M, O) egy PNS probléma folyamat gráfja, $M = \{X_1, \ldots, X_k\}$, és $O = \{u_1, \ldots, u_n\}$. Legyen továbbá

$$O(X_j) = \{u : \ u = (\alpha, \beta) \in O \ \&\ X_j \in \alpha\}$$

minden $X_j \in M$-re. Tetszőleges $j \in \{1, \ldots, k\}$-ra legyen

$$A_j = \{\delta : \delta \in \Omega_{\mathbf{M}}^{\max} \ \&\ X_j \in mat^{in}(op(\delta)) \setminus (mat^{out}(op(\delta)) \cup R)\}.$$

A_j azon maximális döntési leképezéseket tartalmazza, melyek X_j miatt nem elégítik ki az $(\mathcal{A}'2)$-t. Minden $\emptyset \neq I \subseteq \{1, ..., k\}$-ra vezessük be az $A_I = \cap_{i\in I} A_i$ jelölést, továbbá legyen $A_\emptyset = \Omega_{\mathbf{M}}^{\max}$. Világos, hogy $I = \{i_1, \ldots, i_l\}$ esetén

$$A_I = \left\{\delta : \delta \in \Omega_{\mathbf{M}}^{\max} \ \&\ \{X_{i_1}, \ldots, X_{i_l}\} \subseteq mat^{in}(op(\delta)) \setminus (mat^{out}(op(\delta)) \cup R)\right\}$$

azon döntési leképezéseket jelenti, amelyek esetén ($\mathcal{A}'2$) éppen az I-beli indexű anyagok miatt sérül. Így a Szitaformulát alkalmazva, az ($\mathcal{A}'2$)-nek megfelelő maximális konzisztens döntési leképezések számára

$$|\Omega_{\mathbf{M}}^{\max} \setminus (A_1 \cup A_2 \cup ... \cup A_k)| = \sum_{I \subseteq \{1,...k\}} (-1)^{|I|} \cdot |A_I|$$

adódik.

5.1.1. Megjegyzés. Figyeljük meg, hogy a kapott korlát független az előállítandó anyagok halmazától, azaz érvényes bármely $P \subseteq M \setminus R$-re.

5.1.1. Példa. *Az általános esetre vonatkozó korlátszámítás szemléltetésére legyen* $M = \{X_1, \ldots, X_{12}\}$, $O = \{u_1, \ldots, u_7\}$, $P = \{X_8\}$ *és* $R = \{X_{10}, X_{11}, X_{12}\}$, *ahol a műveleti egységek bemeneti és kimeneti anyagait a 5.1. táblázat tartalmazza.*

Műveleti egységek

	bemenetek	kimenetek
u_1	X_{10}	X_1, X_2
u_2	X_{11}	X_3, X_4, X_5
u_3	X_{12}	X_5, X_6
u_4	X_1	X_2, X_8
u_5	X_2, X_3	X_7, X_8
u_6	X_5, X_6	X_8, X_9
u_7	X_6	X_5, X_8

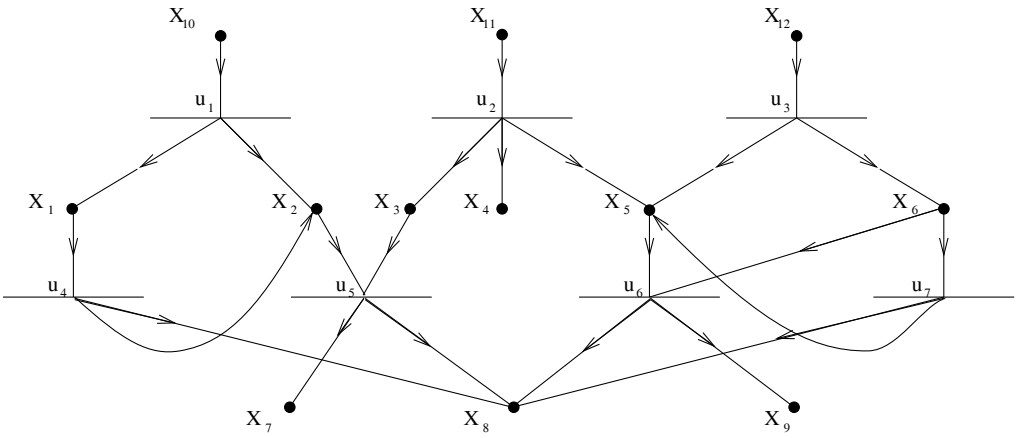
5.1. táblázat.

A megfelelő folyamat gráfot az 5.1. ábra szemlélteti.

A maximális konzisztens döntési leképezések és O *részhalmazai közötti kapcsolatot felhasználva tudjuk, hogy* A_1 *akkor és csakis akkor tartalmazza a* δ*-t, ha* $op(\delta)$ *kielégíti az* $u_1 \notin op(\delta)$ *és* $u_4 \in op(\delta)$ *tulajdonságokat. Ezen maximális döntési leképezések száma* 2^5, *tehát* $|A_1| = 2^5$. *Hasonlóan kapjuk, hogy* $|A_2| = 2^4$, $|A_3| = 2^5$, $|A_5| = 2^3$, $|A_6| = 3 \cdot 2^4$ *és* $|A_j| = 0$ *a többi* j *indexre. Következésképpen*

$$\sum_{I \subseteq \{1,\ldots,k\} \ \& \ |I|=1} |A_I| = 136.$$

A kételemű részhalmazokat vizsgálva, $A_{\{1,2\}}$ *akkor és csakis akkor tartalmazza* δ*-t, ha* $u_1, u_4 \notin op(\delta)$ *és* $u_4, u_5 \in op(\delta)$, *ezért* $A_{\{1,2\}} = \emptyset$. *Hasonlóan* $|A_{\{1,3\}}| = 2^3$, *mivel* $A_{\{1,3\}}$ *akkor és csakis akkor tartalmazza* δ*-t, ha*



5.1. ábra.

$u_1, u_2 \notin op(\delta)$ és $u_4, u_5 \in op(\delta)$. Kiszámolva és összegezve a részhalmazokra a megfelelő értékeket azt kapjuk, hogy:

$$\sum_{I \subseteq \{1,\ldots,k\} \ \& \ |I|=2} |A_I| = 60.$$

Folytatva az eljárást, a három elemű részhalmazokra a 12 értéket kapjuk. Végül azt tapasztaljuk, hogy $|I| > 3$-ra $|A_I| = 0$. Következésképpen a keresett érték:

$$2^7 - 136 + 60 - 12 = 40.$$

Megjegyezzük, hogy ebben a példában $|\Omega_{\mathbf{M}}^{\max}| = 128$ és $|S(\mathbf{M})| = 19$. Az $|S(\mathbf{M})|$ kiszámításához megjegyezzük, hogy a műveleti egységek egy részhalmaza által meghatározott struktúra csak akkor lehetséges megoldás, ha az $(\mathcal{A}2)$ tulajdonság mellett még $(\mathcal{A}1)$ és $(\mathcal{A}3)$ is teljesül. Az $(\mathcal{A}4)$ ilyenkor automatikusan igaz. Az $(\mathcal{A}1)$ miatt vigyázni kell arra, hogy a $P = \{X_8\}$ előállítódjon. Az $(\mathcal{A}3)$ miatt arra kell ügyelnünk, hogy bármely kiválasztott géptől elérhető legyen egy irányított úton az X_8 anyag a folyamatgráfban. Tehát pl. az $\{u_1, u_4\}$ által meghatározott struktúra lehetséges megoldás, az $\{u_2, u_3\}$ nem elégíti ki az $(\mathcal{A}1)$ feltételt. Lehetséges megoldás az $\{u_1, u_2, u_3, u_4, u_6, u_7\}$ gépek által meghatározott struktúra, azonban az $\{u_1, u_2, u_3, u_4, u_5\}$ nem, mert az $(\mathcal{A}3)$ nem teljesül, ugyanis az u_3 által előállított anyagokra, az X_5, X_6-ra egyik gépnek sincs szüksége, tehát u_3 közvetve sem vehet részt a kívánt anyag előállításában. Az $\{u_2, u_3, u_4, u_5, u_6, u_7\}$ géphalmaz viszont még az $(\mathcal{A}2)$-nek sem felel meg, hiszen az u_4 számára fontos lenne bemenetként az X_1 anyag, viszont ezt egyik kiválasztott gép sem gyártja. Ez utóbbi műveleti egység halmazhoz tartozó döntésleképezés tehát nem maximális konzisztens döntésleképezés, tehát még a 40 megszámolt között sincs.

A Szitaformulában $|A_I|$ meghatározására van szükség, ami általában,

tetszőleges folyamat gráf esetén, rendkívül bonyolult: $|A_I|$ az $\{X_{i_1}, \ldots, X_{i_l}\}$ azon $\alpha_{j_1}, \ldots, \alpha_{j_s}$ fedőrendszereinek számával egyenlő, amelyekre léteznek az $(\alpha_{j_t}, \beta_{j_t}) \in O$, $t = 1, \ldots, s$ műveleti egységek úgy, hogy $\{X_{i_1}, \ldots, X_{i_l}\} \cap \beta_{j_t} = \emptyset$, $t = 1, \ldots, s$-re.

5.2. Speciális esetek

Bizonyos sajátos esetekben $|A_I|$ meghatározása természetesen egyszerűsödhet. A továbbiakban azt a speciális esetet fogjuk megvizsgálni, amikor egyetlen input anyaggal működő, ún. szeparátor típusú műveleti egységeink vannak, melyekre tehát $|\alpha| = 1$, bármely $u = (\alpha, \beta) \in O$ műveleti egységre. Legyen ismételten $I = \{i_1, \ldots, i_l\}$, és

$$O^*(X_{i_j}) = O(X_{i_j}) \setminus (\cup_{i \in I} \Delta(X_i)).$$

$O^*(X_{i_j})$ azon műveleti egységek halmaza, melyeknek bemeneti anyaguk az X_{i_j}, de nem termelnek egyetlen anyagot sem az $\{X_t : \ t \in I\}$ halmazból. Akkor $|A_I|$-re a következő képletet kapjuk ([5]):

$$|A_I| = \left(\prod_{t=1}^{l} \left(2^{|O^*(X_{i_t})|} - 1 \right) \right) \cdot 2^{|O \setminus (\cup_{i \in I} \Delta(X_i)) \setminus (\cup_{i \in I} O(X_i))|}.$$

A továbbiakban két speciális szeparátor típusú műveleti egységeket tartalmazó PNS problémaosztály esetén a lehetséges megoldás struktúrák számára explicit módon kiszámolható képleteket fogunk adni.

Mindkét esetben legyen $M = \{X_1, \ldots, X_k\}$ az anyagok halmaza és $O = \{u_1, \ldots, u_k\}$ a műveleti egységek halmaza. Figyeljük meg tehát, hogy az anyagok és műveleti egységek száma egyenlő.

Az első, az ún. Egyenes modellben

$$u_1 = (\alpha_1, \beta_1), \text{ ahol } \alpha_1 = X_1 \text{ és } \beta_1 = X_2,$$
$$u_k = (\alpha_k, \beta_k), \text{ ahol } \alpha_k = X_k \text{ és } \beta_k = X_{k-1},$$

és általában:

$$u_i = (\alpha_i, \beta_i), \text{ ahol } \alpha_i = X_i \text{ és } \beta_i = \{X_{i-1}, X_{i+1}\}, (2 \leq i \leq k-1).$$

Elképzelhetjük sorban egymás mellett a műveleti egységeket úgy, hogy mindegyik egyetlen bemeneti anyaggal rendelkezik, és a két szomszédjának a bemeneti anyagait gyártja. Gondolhatunk itt anyag helyett pl. információra is.

Az egyenes két végén csak egyetlen szomszéd van. Ha teljesebb szimmetriát akarunk, akkor a második, a Lánc modellünket is tekinthetjük, ahol

$$\beta_1 = \{X_2, X_k\} \text{ és } \beta_k = \{X_{k-1}, X_1\}.$$

Mindkét modellünkben lehetséges $|A_I|$ kiszámítása, azonban számos kombinatorikai probléma adódik, amelyeket meg kell oldanunk. A műveleti egységek részhalmazaiban gondolkodva, nyilván az A_I olyan $S_I \subseteq O$ műveleti egységek halmazának megfelelő maximális konzisztens döntési leképezéseket tartalmaz, melyekre $u_{i_s} \in S_I$, de az u_{i_s} egyik szomszédja sincs S_I-ben ($1 \leq s \leq j$). (Az Egyenes modellben a két szélső műveleti egységnek csak egy szomszédja van, a Lánc modellben mindegyik műveleti egységnek egységesen két szomszédja van.) Adott I-re legyen tehát $S_I = \{u_{i_1}, \ldots, u_{i_j}\}$, $N_I(i)$ az u_i szomszédai indexeinek halmaza, és jelölje

$$F_I = \{i' : \ i' \neq i_s \text{ és } i' \notin N(i_s),\ 1 \leq s \leq j\ , 1 \leq i' \leq k\}$$

a "szabad" műveleti egységek halmazát, tehát azokét, amelyek működése nem érinti ($\mathcal{A}'2$) teljesülését. Egy konkrét I -re $|A_I| = 2^{|F_I|}$. Sajnos $|F_I|$ nem csak pusztán j-től függ, hanem az I struktúrájától is. Az $u_{i_1}, ..., u_{i_j}$ nem szomszédos műveleti egységek az Egyenes vagy a Lánc modellben. A leszámolás szempontjából nagyon fontos, hogy hány olyan intervallumra vágják fel a műveleti egységeket, amely egyelemű, és ezek hogyan helyezkednek el.

5.3. Az Egyenes modell

Tegyük fel, hogy az $S_I = \{u_{i_1}, \ldots, u_{i_j}\}$ műveleti egységei r olyan intervallumot határoznak meg I-ben, amelyek 1 hosszúak. Egy intervallum hosszúságán az I által meghatározott műveleti egység halmaz két eleme közötti műveleti egységek számát értjük. Három esetet különböztethetünk meg az I típusa alapján:

a) $i_1 = 1$ és $i_j = k$,

b) $i_1 = 1$ és $i_j < k$ vagy $i_1 > 1$ és $i_j = k$, és

c) $i_1 > 1$ és $i_j < k$.

Könnyű belátni hogy ezen eseteken belül F_I elemszáma azonos, rendre $|F_I| = k - 3j + r + 2$, $|F_I| = k - 3j + r + 1$, illetve $|F_I| = k - 3j + r$.

Kérdés, hogy hány olyan I halmaz van, mely a fenti eseteknek megfelel. Jelölje $L_{[]}(r, j, k)$, $L_{[)}(r, j, k)$, illetve $L_{()}(r, j, k)$ a fenti eseteknek megfelelő I-k számát. Az 1 hosszú (r darab) műveleti egység intervallumokat $\binom{j-1}{r}$ módon lehet kiválasztani, míg a maradék műveleti egységeket a hosszabb intervallumokba rendre $\binom{k-2j}{j-r-2}$, $\binom{k-2j}{j-r-1}$, $\binom{k-2j}{j-r}$-féleképpen lehet szétosztani a három esetben. Következésképpen

$$L_{[]}(r, j, k) = \binom{j-1}{r} \cdot \binom{k-2j}{j-r-2},$$

$$L_{[)}(r, j, k) = \binom{j-1}{r} \cdot \binom{k-2j}{j-r-1}, \text{ és}$$

$$L_{()}(r, j, k) = \binom{j-1}{r} \cdot \binom{k-2j}{j-r}.$$

Figyelembe véve a paraméterek határait, a Szitaformula az Egyenes modell esetében azt adja, hogy ([6]):

$$L^{(1)} = 2^k + \sum_{1 \le j \le \frac{k+1}{2}} (-1)^j \cdot \left[\sum_{\substack{0 \le r \le j-1 \\ k-3j+r+2 \ge 0}} \binom{j-1}{r} \cdot \binom{k-2j}{j-r-2} \cdot 2^{k-3j+r+2} + \right.$$

$$+ \sum_{\substack{0 \le r \le j-1 \\ k-3j+r+1 \ge 0}} \binom{j-1}{r} \cdot \binom{k-2j}{j-r-1} \cdot 2^{k-3j+r+1} +$$

$$\left. + \sum_{\substack{0 \le r \le j-1 \\ k-3j+r \ge 0}} \binom{j-1}{r} \cdot \binom{k-2j}{j-r} \cdot 2^{k-3j+r} \right].$$

5.4. A Lánc modell

Jelen esetben is az Egyenes modell jelöléseit fogjuk használni. Most a szimmetriának köszönhetően $|F_I|$-t könnyebb kiszámolni, nem szükséges eseteket megkülönböztetni, hanem minden I-re $|F_I| = 2^{k-3j+r}$, az I halmazok számának meghatározása azonban éppen a forgásszimmetria miatt nehezebb. A műveleti egységek 1 hosszú intervallum struktúráinak száma:

$$\left(\begin{array}{c} j \\ r \end{array} \right) \cdot \left(\begin{array}{c} k-2j-1 \\ j-r-1 \end{array} \right),$$

míg az $\{u_1; \text{struktúra}\}$ párok száma:

$$k \cdot \left(\begin{array}{c} j \\ r \end{array} \right) \cdot \left(\begin{array}{c} k-2j-1 \\ j-r-1 \end{array} \right),$$

de így j-szer számoltuk az I -ket, ugyanis az u_1-gyel j-féleképpen vághatjuk el a láncot, bármely intervallumban, így a párok száma j-szer több. A Szitaformulával megkapjuk a keresett számot a Lánc modell esetében is ([6]):

$$C^{(1)} = 2^k + \sum_{1 \le j < \frac{k}{2}} (-1)^j \cdot \sum_{\substack{0 \le r \le j-1 \\ k-3j+r \ge 0}} \frac{k}{j} \cdot \left(\begin{array}{c} j \\ r \end{array} \right) \cdot \left(\begin{array}{c} k-2j-1 \\ j-r-1 \end{array} \right) \cdot 2^{k-3j+r} + e_k,$$

ahol

$$e_k = \begin{cases} (-1)^{\frac{k}{2}} \cdot 2 & \text{, ha } k \text{ páros,} \\ 0 & \text{, ha } k \text{ páratlan.} \end{cases}$$

Az e_k „hibatag” kezeli a maximális j esetét külön, ilyenkor csak két I van, a páratlan indexek, vagy a páros indexek.

A fentiek alapján elmondhatjuk, hogy az $L^{(1)}$ és $C^{(1)}$ formulák a tekintett két modellben az $|S(\mathbf{M})|$-et felülről korlátozzák.

5.4.1. Azonosságok

Általános esetben a felmerülő bonyolult kombinatorikai és gráfelméleti problémák következtében nem várható, hogy a Szitaformula kiszámolható korlátot adjon a lehetséges megoldás struktúrák számára. A tárgyalt speciális struktúrák esetén azonban olyan képleteket kaptunk, melyek lehetővé teszik a felső korlátok közvetlen kiszámolását.

Először az Egyenes modellben, egy $U \subseteq O$, $U = op(\delta)$ akkor teljesíti az $(\mathcal{A}'2)$ feltételt, ha minden $u_i \in U$ műveleti egységnek van szomszédja U-ban. Legyen $O = \{u_1, ..., u_k\}$ és $U = \{u_{i_1}, ..., u_{i_t}\}$. Az U-beli műveleti egységek q darab 1 hosszú intervallumot határoznak meg. A fentiekhez hasonló módon megszámlálhatjuk az U és $O \setminus U$ partícióit. Akkor ([6]):

$$L^{(2)} = 1 + \sum_{2 \le t \le k} \sum_{1 \le q \le \min\{\frac{t}{2}; k-t+1\}} \binom{t-q-1}{q-1} \cdot \binom{k-t+1}{q}.$$

Az U részhalmazainak közvetlen megszámlálása a Lánc modellben sokkal bonyolultabb, ebben az esetben a szimmetria nem segít. A Láncban az u_1 helyzete szerint 3 esetet különböztetünk meg:

1) $u_1 \in U$ és bal oldalán van egy másik U-beli elem, de jobb oldalán nincsen;

2) $u_1 \in U$ és az ő $(i-1)$ jobboldali szomszédja U-beli elem, ahol $(i > 1)$;

3) $u_1 \notin U$ és az ő $(i-1)$ jobboldali szomszédja nem eleme U-nak, ahol $(i \ge 1)$.

Ezek az összes lehetséges különböző esetek. Rögzített (k, t, q, i) paraméterekre az eseteknek megfelelően az U részhalmazainak száma:

$$\binom{t-q-1}{q-1} \cdot \binom{k-t-1}{q-1},$$

$$\binom{t-i-q+1}{q-1} \cdot \binom{k-t-1}{q-1}, \text{ illetve}$$

$$\left(\begin{array}{c} t-q-1 \\ q-1 \end{array} \right) \cdot \left(\begin{array}{c} k-t-i \\ q-1 \end{array} \right).$$

Így a következő képletet kapjuk ([6]):

$$C^{(2)} = 1 + \sum_{2 \le t \le k} \left[\sum_{1 \le q \le \min\{\frac{t}{2}; k-t\}} \left(\begin{array}{c} t-q-1 \\ q-1 \end{array} \right) \cdot \left(\begin{array}{c} k-t-1 \\ q-1 \end{array} \right) + \right.$$

$$+ \sum_{2 \le i \le t} \sum_{1 \le q \le \frac{t-i}{2}+1} \left(\begin{array}{c} t-i-q+1 \\ q-1 \end{array} \right) \cdot \left(\begin{array}{c} k-t-1 \\ q-1 \end{array} \right) +$$

$$\left. + \sum_{1 \le i \le k-t} \sum_{1 \le q \le \min\{\frac{t}{2}; k-t-i+1\}} \left(\begin{array}{c} t-q-1 \\ q-1 \end{array} \right) \cdot \left(\begin{array}{c} k-t-i \\ q-1 \end{array} \right) \right] + 1,$$

melyben az első egyes a $t = 0$, az utolsó egyes pedig a $t = k$ esetet jelenti.

Összesítve, két szép kombinatorikus azonosságot kapunk ([6]):

$$L^{(1)} = L^{(2)} \quad \text{és} \quad C^{(1)} = C^{(2)}.$$

5.4.2. A korlátok tulajdonságai

Az eredmény hatékonyságának szemléltetésére az alábbi táblázatokban leírjuk az $|S(\mathbf{M})|$-re kapott korlátokat. Míg az első táblázat a különböző méretű, Egyenes és Lánc struktúrájú feladatokra kiszámolt korlátokat tartalmazza, a második táblázat a korlátok és a maximális döntési leképezések számának arányait szemlélteti a különböző méretű feladatokra.

k	$L^{(1)}$	$C^{(1)}$
3	4	5
4	7	10
5	12	17
6	21	29
7	37	51
8	65	90
9	114	158
10	200	277
11	351	486
12	616	853
13	1081	1497
14	1897	2627
15	3329	4610
16	5842	8090
17	10252	14197
18	17991	24914
19	31572	43721
20	55405	76725
21	97229	134643
22	170625	236282
23	299426	414646
24	525456	727653
25	922111	1276942
26	1618192	2240877
27	2839729	3932465
28	4983377	6900995
29	8745217	12110402
30	15346786	21252274

k	$\frac{L^{(1)}}{2^k}$	$\frac{C^{(1)}}{2^k}$
3	5.0000000000E-01	6.2500000000E-01
4	4.3750000000E-01	6.2500000000E-01
5	3.7500000000E-01	5.3125000000E-01
6	3.2812500000E-01	4.5312500000E-01
7	2.8906250000E-01	3.9843750000E-01
8	2.5390625000E-01	3.5156250000E-01
9	2.2265625000E-01	3.0859375000E-01
10	1.9531250000E-01	2.7050781250E-01
11	1.7138671875E-01	2.3730468750E-01
12	1.5039062500E-01	2.0825195313E-01
13	1.3195800781E-01	1.8273925781E-01
14	1.1578369141E-01	1.6033935547E-01
15	1.0159301758E-01	1.4068603516E-01
16	8.9141845705E-02	1.2344360352E-01
17	7.8216552731E-02	1.0831451416E-01
18	6.8630218504E-02	9.5039367684E-02
19	6.0218811040E-02	8.3391189581E-02
20	5.2838325502E-02	7.3170661930E-02
21	4.6362400052E-02	6.4202785490E-02
22	4.0680170050E-02	5.6334018708E-02
24	3.1319618239E-02	4.3371498586E-02
26	2.4112939849E-02	3.3391669387E-02
28	1.8564525990E-02	2.5708209726E-02
30	1.4292808268E-02	1.9792722558E-02
40	3.8662157089E-03	5.3539468199E-03
60	2.8289357042E-04	3.9175187251E-04
80	2.0700221556E-05	2.8661325825E-05
100	1.5277899843E-06	2.0515052235E-06

Sajnos a korlátok még ebben a két speciális esetben sem mondhatók elég jónak. Össze lehet őket hasonlítani P ismeretében a lehetséges megoldás struktúrák számával, de a P nélkül csak a maximális döntési leképezések számával. Mindenesetre a fenti táblázatokból kiolvasható, hogy a korlát és a maximális döntési leképezések számának aránya a k növekedésével gyorsan csökken.

6. fejezet

A PNS bottleneck és k-összeg változatai

A súlyozott PNS probléma, amikor a célfüggvény a megoldásban részt vevő műveleti egységek súlyainak összege (4.1) alakú, speciális Minsum probléma. Megmutatjuk, hogy a PNS probléma k-összeg változata jól-megoldható, így az általános Minsum problémák abba az osztályába tartozik, amelyekre a Minsum változat NP-teljes, míg a k-sum változat jól-megoldható rögzített k esetén [8].

6.1. Minsum, Bottleneck és k-összeg optimalizálási problémák

Legyen E egy véges halmaz, F az E részhalmazainak egy családja, valamint $f : F \to \Re$ olyan függvény, amely minden $S \in F$ részhalmazhoz egy valós számot rendel. Egy általános kombinatorikus optimalizálási probléma olyan $S^* \in F$ elemet találni, melyre $f(S^*) \leq f(S)$, minden $S \in F$ esetén. Számos kombinatorikus optimalizálási feladatban adott egy súlyozás $c : E \to \Re$ az E elemein, és az $f(S)$ célfüggvény ezen súlyok függvényeként van definiálva. Amennyiben $f(S) = \max\{c_e : e \in S\}$, a *Minmax* vagy *Bottleneck Optimalizálási problémát* kapjuk, rövidítve (BOP).

Ha $f(S) = \sum_{e \in S} c_e$, akkor jutunk egy *Minsum problémához*, rövi-

dítve (MSP), ahol c_e jelentse mindig a $c(e)$ értékét az egyszerűbb jelölés kedvéért. Ismert néhány NP-teljes probléma a (BOP) és az (MSP) problémaosztályokon belül. Olyan speciális szerkezetű F is van azonban, amikor létezik hatékony megoldás; ilyen pl. a hozzárendelési probléma.

Egy $S \in F$ részhalmazra legyen feltéve, hogy S elemeit a súlyaik szerint nemnövekvő sorrendben soroljuk fel. Precízebben $S = \{s_1, \ldots, s_{|S|}\}$ és $c_{s_i} \geq c_{s_{i+1}}$, ha $i = 1, \ldots, |S| - 1$. Tetszőleges pozitív k egészre legyen $f_k(S) = \sum_{i=1}^{p} c_{s_i}$, ahol $p = \min\{|S|, k\}$. Ekkor a *k-összeg optimalizálási probléma* (rövidítve SUM(k)) a következő:

$$\min\{f_k(S) : S \in F\}.$$

Ha $k = 1$, a SUM(k) probléma a BOP problémát adja vissza, ha pedig $k \geq \max\{|S| : S \in F\}$, akkor az MSP adódik. Tehát a SUM(k) közös általánosítása a BOP és MSP problémáknak; Ebből következik, hogy akár a BOP akár az MSP NP-teljes, akkor a SUM(k) is az.

Az általános SUM(k) problémát először Gupta és Punnen [29] tanulmányozta, majd később Punnen és Aneja [47]. Megmutatták, hogy a SUM(k) problémát meg lehet úgy oldani, ha megoldunk $O(m)$ darab Minsum problémát, ahol ($m = |E|$). Ha tehát lenne polinomidejű algoritmus a Minsum problémára, akkor a SUM(k) problémát tetszőleges k, ($1 \leq k \leq m$) esetén meg tudnánk hatékonyan oldani. A PNS problémára a Minsum változat NP-teljes [4], tehát nem hívhatjuk segítségül a SUM(k) feladathoz.

6.2. Bottleneck PNS probléma

Legyen adott egy PNS probléma $\mathbf{M} = (P, R, O)$ redukált strukturált modellje. Szeretnénk találni egy olyan lehetséges megoldást, amelyben a legköltségesebb műveleti egység költsége a lehető legkisebb. Formálisan a következőt akarjuk megoldani:

$$\min\{\max\{w(u) : u \in o\} : (m, o) \in S(\mathbf{M})\}. \tag{6.1}$$

A 6.1 optimalizálási feladatban az összes műveleti egység halmaza legyen $O = \{u_1, \ldots, u_n\}$. Feltehető, hogy $w(u_1) \leq w(u_2) \leq \cdots \leq w(u_n)$. Minden pozitív $i(\leq n)$ egészre legyen $O_i = \{u_1, \ldots, u_i\}$ and $M_i = mat(O_i)$. Továbbá legyen $\mathbf{M}_i = (P, R, O_i)$. Ekkor igaz a következő állítás:

6.2.1. Lemma. *Ha valamely $1 \leq i \leq n$ egészre $S(\mathbf{M}_i)$ maximális struktúrája létezik, viszont $S(\mathbf{M}_{i-1})$ maximális struktúrája nem létezik, akkor az u_i benne van az összes $S(\mathbf{M}_i)$-beli lehetséges megoldásban.*

Az előző állításból következik, hogy 6.1 optimális megoldásának értéke $w(u_i)$. Valóban, legyen $(m', o') \in S(\mathbf{M})$ egy tetszőleges lehetséges megoldás. Abban az esetben ha van egy $j > i$ indexű $u_j \in o'$, akkor $w(u_i) \leq \max\{w(u) : u \in o'\}$. Tegyük tehát fel, hogy $u_j \in o'$ esetén $j \leq i$. Ha $u_i \in o'$, akkor $w(u_i) = \max\{w(u) : u \in o'\}$. Végül pedig $u_i \notin o'$ lehetetlen, mivel (m', o') az $\mathbf{M}_i$ egy lehetséges megoldása, tehát a 6.2.1 Lemma szerint, $u_i \in o'$. Következésképpen

$$w(u_i) = \min\{\max\{w(u) : u \in o\} : (m.o) \in S(\mathbf{M})\}.$$

A következő eljárással tehát meg tudjuk oldani az 6.1 feladatot.

1. Eljárás

- *Inicializálás* Legyen $i = 1$ és $O_i = \{u_1\}$.
- *Iteráció (i.)* Legyen $M_i = mat(O_i)$. Alkalmazzuk a maximális struktúra generáló algoritmust az $\mathbf{M}_i = (P, R, O_i)$ strukturális modellre. Ha létezik a maximális struktúra, akkor megállunk; a maximális struktúra tartalmazza az optimális megoldást és az optimum értéke $w(u_i)$. Ellenkező esetben legyen $O_{i+1} = \{u_1, \ldots, u_{i+1}\}$, $i := i + 1$, és folytassuk a következő iterációval.

Könnyű látni, hogy az eljárás időfelhasználása legfeljebb $n \cdot q(n)$, ahol $q(n)$ jelöli a maximális struktúra generáló algoritmus maximális időigényét.

6.2.1. Megjegyzés. *A fenti időbonyolultság a szokásos módon javítható. Ha az eljárásban az $i = [n/2]$ értékkel kezdünk és azt találjuk, hogy létezik maximális struktúra, akkor a következő lépésben válasszuk az $[1, i]$ intervallum egyik középső pontját, illetve ellenkező esetben az $[i, n]$ intervallum egyik középső pontját. Ezzel az ismert módszerrel az algoritmus futási idejét $q \cdot \log(n)$-nel becsülhetjük.*

6.3. A PNS probléma k-összeg változata

Legyen (M, O) a maximális struktúrája az $\mathbf{M} = (P, R, O)$ PNS probléma redukált strukturális modelljének. Továbbá legyen k egy rögzített pozitív egész. Olyan lehetséges megoldást szeretnénk találni, amelyre a k legköltségesebb műveleti egység költségének összege minimális.

Formalizálva a problémát, legyen $u_{i_1}, \ldots, u_{i_k}$, a k legköltségesebb műveleti egysége az (m, o) lehetséges megoldásnak. A *k-összeg* PNS *probléma* ekkor

$$\min\{\sum_{t=1}^{k} w(u_{i_t}) : (m, o) \in S(\mathbf{M})\}, \tag{6.2}$$

ahol ha egy lehetséges megoldás kevesebb gépet tartalmaz mint k, akkor fiktív, 0 súlyú műveleti egységekkel egészítjük ki a valódiakat. Mivel az o egyértelműen meghatározza az (m, o) P-gráfját minden lehetséges megoldásra, (6.2) valóban egy speciális SUM(k) probléma.

A (6.2) megoldása érdekében, legyen $O = \{u_1, \ldots, u_n\}$. Az általánosság megszorítása nélkül most is feltehetjük, hogy $w(u_1) \leq w(u_2) \leq \cdots \leq w(u_n)$. Rendezzük most még az O legfeljebb k elemű részhalmazait is (a lineáris rendezés jele legyen $\preceq$) a következő módon

$$\{u_{i_1}, \ldots, u_{i_r}\} \preceq \{u_{j_1}, \ldots, u_{j_s}\} \text{ akkor és csak akkor, ha}$$

$$\sum_{t=1}^{r} w(u_{i_t}) \leq \sum_{t=1}^{s} w(u_{i_t}).$$

Ilyen rendezés létezik, további szabályokkal egyértelműsíthető.

Minden $\{u_{i_1}, \ldots, u_{i_r}\} \subseteq O$ részhalmazhoz legyen

$$O_{\{u_{i_1}, \ldots, u_{i_r}\}} = \{u_{i_1}, \ldots, u_{i_r}\} \cup \{u_t : u_t \in O \;\&\; w(u_t) \leq w(u_{i_1})\},$$

ahol feltesszük, hogy u_{i_1} az $\{u_{i_1}, \ldots, u_{i_r}\}$ részhalmazban előforduló legkisebb indexű elem. Legyen továbbá $\mathbf{M}_{\{u_{i_1}, \ldots, u_{i_r}\}} = (P, R, O_{\{u_{i_1}, \ldots, u_{i_r}\}})$. Ekkor a következő állítás teljesül.

6.3.1. Tétel. *Ha a műveleti egységek valamely* $\{u_{i_1}, \ldots, u_{i_r}\} \subseteq O$ *részhalmazára az* $S(\mathbf{M}_{\{u_{i_1},\ldots,u_{i_r}\}})$ *maximális struktúrája létezik, és minden* $\{u_{j_1}, \ldots, u_{j_s}\} \subseteq O$ *olyan részhalmazra, melyre* $\{u_{j_1}, \ldots, u_{j_s}\} \preceq \{u_{i_1}, \ldots u_{i_r}\}$ *teljesül, valamint* $S(\mathbf{M}_{\{u_{j_1},\ldots,u_{j_s}\}})$ *maximális struktúrája nem jön létre, akkor* $S(\mathbf{M}_{\{u_{i_1},\ldots,u_{i_r}\}})$ *maximális struktúrája egy optimális megoldása (6.2)-nek, és az optimum értéke* $\sum_{t=1}^{r} w(u_{i_t})$.

A 6.3.1 Tétel alapján a következő eljárás megad egy optimális megoldást.

2. Eljárás

1. lépés Állítsuk elő a megfelelő lineáris rendezést.

2. lépés Legyen $i = 1$.

3. lépés Tekintsük az O i-edik részhalmazát a rögzített rendezés szerint. Legyen $\{u_{i_1}, \ldots, u_{i_r}\}$ ez a részhalmaz. Futtassuk le a maximális struktúra generáló algoritmust az $\mathbf{M}_{\{u_{i_1},\ldots,u_{i_r}\}}$ modellre. Ha létezik maximális struktúra, akkor az eljárás befejeződött; a maximális struktúra egy optimális megoldás. Egyébként legyen $i := i + 1$ és ismételjük a 3. lépést.

Az eljárás időkomplexitása $\sum_{t=1}^{k} \binom{n}{t} \cdot q(n)$ ahol $q(n)$ jelöli itt is a maximális struktúra generáló eljárás időkomplexitását.

Érdemes megjegyezni, hogy a fent bemutatott technika alkalmas 6.2-nél még általánosabb probléma megoldására is. Nevezetesen ha a célfüggvény nem a k legdrágább műveleti egység költségeinek az összege, hanem csak éppen ezektől függ, vagyis *pl.*, $z(w(u_{i_1}), \ldots, w(u_{i_k}))$ alakú; ahol fiktív gépeket megengedünk, akkor a következő problémát kapjuk:

$$\min\{z(w(u_{i_1}), \ldots, w(u_{i_k})) : (m, o) \in S(\mathbf{M})\}. \tag{6.3}$$

Természetesen a lineáris rendezés most a következő alakot ölti:

$$\{u_{i_1}, \ldots, u_{i_r}\} \preceq \{u_{j_1}, \ldots, u_{j_s}\} \text{ akkor és csak akkor, ha}$$

$$z(w(u_{i_1}), \ldots, w(u_{i_k})) \leq z(w(u_{j_1}), \ldots, w(u_{j_k})).$$

A bottleneck PNS-probléma jól megoldható, míg a PNS probléma NP-teljes ([4], [24], [37]). Módszerünk a k-összeg változatra természetesen csak rögzített (kicsi) k esetén polinomiális. Hasonlóan pl. az ütemezés elméletéből vett k-késés összegének optimalizálása problémához, amint az Woeginger cikkében [50] olvasható.

7. fejezet

Egyszerűsített PNS problémák

Minden PNS problémát egy egyszerűbb műveleti egységeket tartalmazó PNS feladatba lehet transzformálni. Ez az észrevétel lehetővé teszi a súlyozott gráfokra vonatkozó híres éllefedési probléma egy szép alkalmazását PNS feladatok közelítő megoldásának megtalálására. A minimális költségű éllefedési algoritmus (Minimum Cost Edge Cover–MCEC) segítségével új heurisztikus algoritmusokat definiáltunk, és vizsgáltunk meg egyszerűsített PNS feladatokra [11], [12].

Valós problémák esetén is gyakori, hogy a műveleti egységek bemeneti és kimeneti anyaghalmazai kis elemszámúak. Ha tudnánk, hogy csak egyszerű műveleti egységek fordulnak elő, talán ki lehetne használni ezt a tényt. Ebben a részben csak olyan PNS feladatokat tekintünk, amelyek ebben az értelemben egyszerűek. Ezekre dolgozunk ki heurisztikus megoldó algoritmusokat. Ennek érdekében bevezetünk egy olyan műveleti egységek közötti helyettesítési módot, amely egyszerű egységeket használ. Az átalakítás során ügyelünk arra, hogy ne veszítsünk el lehetséges megoldásokat.

A továbbiakban akkor mondjuk, hogy egy műveleti egység *egyszerű*, ha a bemeneti és kimeneti anyagok száma együttesen legfeljebb 3. Alapvető észrevétel, hogy minden műveleti egység helyettesíthető egy egyszerű egységekből álló hálózattal. Ez a megfigyelés lehetővé teszi egy tetszőleges PNS feladathoz történő, vele ekvivalens PNS feladat hozzárendelését, amely csak egyszerű műveleti egységeket tartalmaz. Az ilyen feladatot egyszerűsített PNS feladatnak nevezzük. A hozzárendelés során néhány új műveleti egységet kell definiálni, be kell vezetni új anyagokat is. A feladat méretének növe-

kedése azonban nem olyan mértékű, hogy az elnevezésünket megkérdőjelezze.

7.1. Műveleti egységek helyettesítése kisebbekkel

Legyen $\mathbf{M} = (P, R, O)$ egy tetszőleges PNS feladat és legyen $u = (\alpha, \beta) \in O$ egy tetszőleges műveleti egység az $\alpha = \{X_1, \ldots, X_n\}$ és $\beta = \{Y_1, \ldots, Y_m\}$ anyaghalmazokkal. Ha egy u műveleti egység nem egyszerű, vagyis n=2 és m=2, vagy $n > 2$ és $m > 2$ közül legalább az egyik teljesül, akkor definiáljuk az $u_1, \ldots, u_{n+m}$ gépeket a következőképpen

$$u_1 = (\{X_1, X_2\}, \{Z_1\}),$$

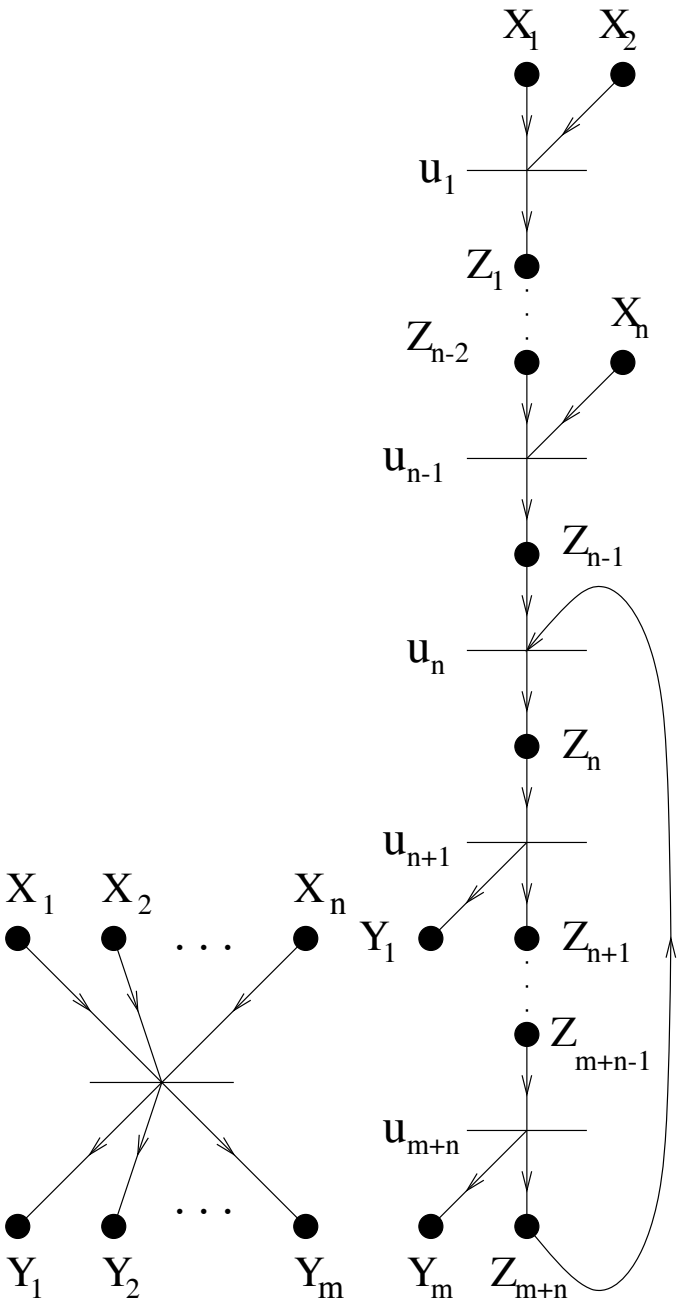
$$u_i = (\{X_{i+1}, Z_{i-1}\}, \{Z_i\}) \qquad i = 2, 3, ..., n-1,$$

$$u_n = (\{Z_{n-1}, Z_{n+m}\}, \{Z_n\}),$$

$$u_{n+j} = (\{Z_{n-1+j}\}, \{Z_{n+j}, Y_j\}) \qquad j = 1, 2, ..., m,$$

ahol $Z_1, Z_2, \ldots, Z_{n+m}$ új anyagokat jelöl, tehát $Z_1, Z_2, \ldots, Z_{n+m} \notin M \cup O$. Helyettesítsük az u műveleti egységet az $u_1, u_2, \ldots, u_{n+m}$ egységekkel az O halmazban, tehát legyen $O^* = O \backslash \{u\} \cup \{u_1, u_2, \ldots, u_{n+m}\}$ az új halmaza a műveleti egységeknek, és a $\{Z_1, Z_2, \ldots, Z_{n+m}\}$ új anyagokat is vegyük hozzá az M-hez. Ismételjük meg a fenti helyettesítést az összes nem egyszerű gépre, ügyelve arra, hogy mindig újabb és újabb egyszerű műveleti egységeket és anyagokat definiáljunk, vagyis legyenek páronként diszjunktak a hozzárendelt $\{u_1, u_2, \ldots, u_{n+m}\}$ műveleti egység halmazok is és a $\{Z_1, Z_2, \ldots, Z_{n+m}\}$ anyaghalmazok is. Legyen $\mathbf{M}^* = (P^*, R^*, O^*)$ a helyettesítések után kapott új PNS feladat, ahol a kívánt anyagok halmaza megmarad, $P^* = P$ és a nyersanyagok halmaza is változatlan lesz ($R^* = R$). Bármely $u \in O$ műveleti egységre legyen $\pi(u) = \{u_1, u_2, \ldots, u_{n+m}\}$ ha u nem volt egyszerű és $\pi(u) = \{u\}$ ha u egyszerű volt. Továbbá a gépek részhalmazaira is vezessük be a logikus jelölést, ha $o \subset O$, legyen $\Pi(o) = \cup\{\pi(u) : u \in o\}$. A fenti egyszerűsítésről a következőt állíthatjuk.

7.1.1. Tétel. *Létezik egy bijektív megfeleltetés $S(\mathbf{M})$ és $S(\mathbf{M}^*)$között.*



7.1. ábra. Egy műveleti egység helyettesítése egyszerűekkel.

Bizonyítás Először megadunk, egy φ leképezést $S(\mathbf{M})$-ből $S(\mathbf{M}^*)$-be és bebizonyítjuk róla, hogy injektív és szürjektív. Minden $(m,o) \in S(\mathbf{M})$ lehetséges megoldásra legyen $\varphi((m,o)) = (m^*,o^*)$, ahol $o^* = \Pi(o)$ és $m^* = mat^{in}(o^*) \cup mat^{out}(o^*)$. Legyen $(m,o) \in S(\mathbf{M})$ egy tetszőleges lehetséges megoldás. Megmutatjuk, hogy ekkor $\varphi((m,o))$ az $\mathbf{M}^*$ egy lehetséges megoldása. Világos, hogy $\varphi((m,o))$ egy P-gráf, amely (M^*,O^*) részgráfja. Következésképpen elegendő ellenőrizni, hogy $\varphi((m,o))$ kielégíti az $(A1)$ - $(A4)$ feltételeket. Az $(A1)$ teljesül, mivel $P^* = P \subseteq m \subseteq m^*$. Az $(A2)$ érvényességéhez elég megfigyelni, hogy
$mat^{out}(o) \cap m = mat^{out}(\Pi(o)) \cap m$ és $mat^{out}(\Pi(o)) \backslash m \cap M = \emptyset$.
Az $(A3)$ feltételhez legyen u^* egy tetszőleges műveleti egység o^*ból és legyen Y_0 az a műveleti egység o-ból, amelyre $u^* \in \pi(Y_0)$. Mivel $(m,o) \in S(\mathbf{M})$, létezik egy $[Y_0, Y_n]$ út (m,o)-ban, melyre $Y_n \in P$. A helyettesítés módja miatt ekkor van egy $[u^*, Y_n]$ út (m^*,o^*)-ban, mivel minden $[Y_{i-1}, Y_i, Y_{i+1}]$ részútja $[Y_0, Y_n]$-nek helyettesíthető a $[Y_{i-1}, u_{i_1}, X_{i_1}, u_{i_2}, X_{i_2}, ..., u_{i_r}, Y_{i+1}]$ részúttal, ahol $\{u_{i_1}, u_{i_2}, ..., u_{i_r}\} \subset \pi(Y_i)$ és $X_{i_j} \in mat^{out}(u_{i_j}) \backslash m$. Tehát találtunk egy $[u^*, Y_n]$ utat az (m^*,o^*) részgráfban, a $\varphi((m,o))$ képben, és $Y_n \in P$ is teljesül. Tehát érvényes az $(A3)$ feltétel is. Végül az $(A4)$ is igaz, mert $m^* = mat^{in}(o^*) \cup mat^{out}(o^*)$.

Most igazoljuk, hogy φ egy kölcsönösen egyértelmű leképezés. Vegyünk két különböző lehetséges megoldást, $(m,o) \neq (m',o') \in S(\mathbf{M})$. Mivel a műveleti egységek halmaza különbözik, $o \neq o'$, így az egyszerű műveleti egységek halmazai is eltérnek egymástól, $\Pi(o) \neq \Pi(o')$. Tehát különböző lehetséges megoldások φ melletti képei szintén különbözők. Még precízebben
$\varphi(m,o) = mat^{in}(\Pi(o)) \cup mat^{out}(\Pi(o))$,
$\Pi(o)) \neq mat^{in}(\Pi(o')) \cup mat^{out}(\Pi(o')), \Pi(o')) = \varphi((m',o'))$.

Megmutatjuk, hogy φ ráképezés. Legyen $(m^*,o^*) \in S(\mathbf{M}^*)$ egy tetszőleges lehetséges megoldás. Keressük meg azt az $(m,o) \in S(\mathbf{M})$ lehetséges megoldást, melyre $\varphi((m,o)) = (m^*,o^*)$. Legyen $o = \{v : v \in O$ és $\exists u \in o^*$ ahol $u \in \pi(v)\}$ és legyen $m = M \cap m^*$. Most be fogjuk látni a négy alapvető feltétel teljesülését (m,o)-ra. Az $(A1)$ világos, mivel $P \subseteq M$ és $P \subseteq m^*$. Ha az $(A2)$ tulajdonsággal ellentétben $\exists X \in m$ melyre $X \notin mat^{out}(o)$ és $X \notin R$, akkor $X \in m^*$ és $X \notin mat^{out}(o^*)$, ami ellentmondás. Továbbá $R \cap mat^{out}(o) = \emptyset$ is teljesül, mivel $R \cap mat^{out}(o^*) = \emptyset$. Ennek következtében az $(A2)$ feltétel fennáll.

Legyenek $u \in o$ és $Y_0 \in \pi(u)$ tetszőleges műveleti egységek. Legyen

továbbá $[Y_0, Y_n]$ egy létező út (m^*, o^*)-ban, ahol $Y_n \in P$. Hagyjuk el az $[Y_0, Y_n]$ útból az összes anyagot és gépet ami (m^*, o^*) -ban előfordul. Amennyiben maradnak anyagok az út mentén, mondjuk Y_i, Y_{i+j} követi egymást, akkor írjuk be az $u \in o$ műveleti egységet a két anyag közé, melyre $Y_i \in mat^{in}(\{u\})$ és $Y_{i+j} \in mat^{out}(\{u\})$. Mindig megtehető az ilyen beszúrás az $(A2)$ tulajdonság és a helyettesítés természete miatt. Az így módosított út (m, o)-ban u-tól az $Y_n \in P$ kívánt termékig halad, tehát az $(A3)$ is teljesül.

Az $(A4)$ tulajdonság vizsgálatához vegyük észre, hogy minden $X \in M \cap m^*$ esetén van olyan $u \in o^*$ és $X \in mat^{in}(\{u\}) \cup mat^{out}(\{u\})$, hogy $X \in mat^{in}(\{v\}) \cup mat^{out}(\{v\})$ ahol $u \in \pi(v)$.

Végül meg tudjuk mutatni, hogy $\varphi((m, o)) = (m^*, o^*)$, ha belátjuk még, hogy $\Pi(o) = o^*$. Nyilvánvalóan $\Pi(o) \supseteq o^*$. Megfordítva, ha $u \in o$ akkor $\pi(u) \subseteq o^*$ az $(A2)$ és a helyettesítés módja miatt, következtetésképpen $\Pi(o) \subseteq o^*$. □

7.2. Alkalmazzuk a minimális összsúlyú éllefedési problémát!

Kifejlesztettünk egy heurisztikus algoritmust az egyszerűsített PNS feladatok megoldására. Minden iterációban lesz egy változó kívánt anyagok halmaza; ez kezdetben a P. Az eljárás ehhez a halmazhoz fog mindig egy G gráfot hozzárendelni a következő módon: Két kívánt anyagot pontosan akkor kötünk össze egy éllel, ha létezik olyan műveleti egység, nevezzük *szülőnek*, amely két kimeneti anyaga éppen ez a két anyag. Lehetséges több szülő is.

A G gráf ezen éléhez rendelt költsége legyen a minimális súlyú szülő műveleti egység súlya. Az izolált csúcsokat elhagyva a kapott gráfra alkalmazzuk a minimális súlyú éllefedési algoritmust. A kapott minimális súlyú éllefedésben szereplő élekhez tartozó szülő műveleti egységek bemeneti anyagai, valamint az izolált csúcsokként le nem fedett anyagok közül mindazok a következő iteráció kívánt anyagai lesznek, amelyeket nem gyárt egyetlen már korábban kiválasztott gép sem. Az eljárás akkor ér véget, ha üres lesz a kívánt anyagok aktuális halmaza. Ekkor az egyes iterációs lépésekben kiválasztott műveleti egységek alkotják a lehetséges megoldásunkat.

Mi motiválta algoritmusunkat? A PNS problémára hatékony megoldási eljárásra nincs esélyünk, mivel az NP-nehéz probléma [4]. Jól ki-

választott speciális problémaosztályokra találtunk jó megoldásokat [9]. Általában azonban be kell érnünk heurisztikus megoldó eljárássokkal. Ebben az algoritmusban azonban legalább minden egyes lépésben optimálisan próbáljuk előállítani a kívánt anyagok aktuális halmazát. Erre kitűnő eszköz az MCEC problémára vonatkozó algoritmus. (l. [45] és [46]). Erre a fontos algoritmusra kevés alkalmazást találtunk, sőt az algoritmus matematikai leírásán túl nem találtunk használható megvalósítást sem. Keserű Kornél diplomamunkáját az MCEC algoritmus megvalósítása témájából írta. Az OTDK munkáját is vezetésemmel írta, díjazott lett. Egyéb, nem súlyozott esetekre vonatkozó kehely-típusú párosítási algoritmusok azonban jól ismertek.

7.3. Kehely-típusú algoritmus

Irányítatlan hálózatokkal foglalkozunk. Legyen $G = (N, A, c = (c_{ij}))$ egy irányítatlan hálózat, ahol a c vektor az élekhez rendelt költségeket tartalmazza, $|N| = n$, és $|A| = m$. Párhuzamos éleket megengedünk, hurokéleket nem, és feltesszük, hogy nincsenek a gráfban izolált csúcsok sem. A c_{ij} számok valósak, általában feltesszük, hogy $c_{ij} \geq 0$ is fennáll. Az élek egy $E \subseteq A$ részhalmazáról azt mondjuk, hogy lefedi a G csúcsait, ha minden N-beli csúcs legalább egy E-beli élre illeszkedik.
Ekkor az E lefedő, vagy dominálló élhalmaz G-ben. Egy párosítás pontosan akkor lefedő élhalmaz, ha teljes párosítás. Feltettük hogy nincs G-ben izolált csúcs, ezért az A élhalmaz maga lefedi a G csúcsait. Létezik hatékony kehely-típusú algoritmus, amely megtalál egy minimális költségű élfedést (l. [45]). Ez Edmonds híres párosítási algoritmusára épül, amely bármely gráfra polinomidőben talál maximális párosítást. Egyszerű tény, hogy minden optimális megoldás-élhalmaz csillagokból álló erdőt határoz meg, ($c_{ij} > 0$) esetén (l. [46]).

Definiáljuk a G-beli $x = (x_{ij})$ lefedő vektort a következőképpen:

$$x_{ij} = \begin{cases} 1, & \text{ha le van fedve a } G\text{-beli } (i;j) \text{ él,} \\ 0, & \text{ha nincs lefedve.} \end{cases}$$

A következő egész értékű (bináris) programozási feladatot kapjuk:

$$min\ z(x) = \sum_{(i,j)\in A)} (c_{ij}x_{ij}),$$

feltéve, hogy $x(i) \geq 1$, minden $i \in N$ esetén, és $x_{ij} \in \{0,1\}$, minden $(i,j) \in A$ esetén, ahol $x(i) := \sum_{j,(i,j)\in A)}(x_{ij})$.

Legyen $Y \subseteq N$, melyre $|Y| \geq 3$ és páratlan.
Legyen $Y^+(x) = \sum_{i,\text{ vagy } j\in Y;(i,j)\in A)}(x_{ij})$.

Ha a fenti feltételrendszerben $x_{ij} \in \{0,1\}$ -t $0 \leq x_{ij} \leq 1$ -re relaxáljuk és helyette még felvesszük a

$$Y^+(x) \geq (|Y|+1)/2 \qquad (*)$$

feltételt minden megfelelő Y-ra, akkor egy LP-t kapunk. A (*) feltétel az ún. *fedési kehely egyenlőtlenség*, amely Edmonds *párosítási kehely egyenlőtlenségére* hasonlít. A fenti LP optimális megoldása egy egész vektor lesz, amely egy minimális élledést ad meg G-ben. A továbbiakban az (MCEC) rövidítés jelentse a speciális LP-re vonatkozó hatékony megoldási eljárást.

7.4. Heurisztikák egyszerűsített PNS problémákra

Legyen (P,R,O) egy egyszerűsített PNS probléma strukturális modellje.

R: a nyersanyagok halmaza,

P: a kívánt anyagok halmaza, természetesen $P \cap R = \emptyset$,

$M \supseteq P \cup R$ az összes anyag halmaza,

Minden $u \in O$ műveleti egység $(1,2)$ vagy $(2,1)$ típusú,

Minden $m_q \in M \setminus R$ esetén létezik, (**)

$(\alpha,\beta) \in O$ amelyre $m_q \in \beta$.

7.5. A heurisztikák leírása

7.5.1. Az 1. heurisztika

Legyen (P, R, O) egy, a feltételeket kielégítő PNS probléma strukturális modellje, azaz minden $u \in O$ gép $(1, 2)$ vagy $(2, 1)$ típusú.

RM_i az i-edik lépés elvégzése után gyártani kívánt anyaghalmaz,

PM_i az i-edik lépés után rendelkezésre álló anyaghalmaz,

$G_i(N_i, A_i)$ $(c(e) : e \in A_i)$.

Pontosan akkor kapcsolunk össze két, $m_q, m_r \in RM_{i-1}$ csúcsot egy e éllel, ha létezik $(\alpha, \beta) \in O$ és $m_q, m_r \in \beta \quad (\beta = \{m_q, m_r\})$.

$(\alpha, \beta)_{q,r}$:= azon minimális súlyú gépek egyike, melyek gyártják mind m_q-t és m_r-t, feltéve, hogy létezik ilyen gép.

$c(e) := w(\alpha, \beta)_{q,r}$ ha $e = (m_q, m_r)$,

$A_i := \{e : e = (m_q, m_r)\ m_q, m_r \in RM_{i-1} m_q \neq m_r$
és $\exists(\alpha; m_q, m_r) \in O\}$.

Ha $A_i = \emptyset$ akkor G_i nem létezik.

$N_i := \{$Az A_i-beli élek végpontjai$\}$ $(\subseteq RM_{i-1})$.

Az algoritmus lépésenkénti leírása:

1./ Inicializálás: $O_0 = \emptyset,\ RM_0 = P,\ PM_0 = R,\ i = 0$

2./ Konstruáljuk meg a G_i gráfot a fent leírt módon.
Ha létezik G_i, akkor $\Longrightarrow$ 3. Ha G_i nem létezik, akkor $\Longrightarrow$ 4.

3./ Alkalmazzuk G_i-re az MCEC algorimust, legyen E_i a minimális súlyú lefedő élhalmaz.

$O_i := O_{i-1} \bigcup \{(\alpha, \beta)_{q,r} : (m_q, m_r) \in E_i\}$,
$RM_i := (P \bigcup mat^{in} O_i) \setminus mat^{out} O_i$,
$PM_i := R \bigcup mat^{out} O_i$.
Ha $RM_i = \emptyset$, akkor $O^H := O_i$ a gépek heurisztika által adott megoldási halmaza. Készen vagyunk.
Ha $RM_i \neq \emptyset$ akkor $i := i + 1, \implies 2$.

4./ $u^i :=$ azon minimális súlyú gépek egyike, mely gyártja valamely RM_{i-1}-beli anyagot.
$O_i := O_{i-1} \bigcup u^i$,
$RM_i = (P \bigcup mat^{in} O_i) \setminus mat^{out} O_i$,
$PM_i = R \bigcup mat^{out} O_i$.
Ha $RM_i = \emptyset$, akkor $O^H := O_i$ a gépek heurisztika által adott megoldási halmaza. Készen vagyunk.
Ha $RM_i \neq \emptyset$ akkor $i := i + 1, \implies 2$.

7.5.2. A 2. heurisztika

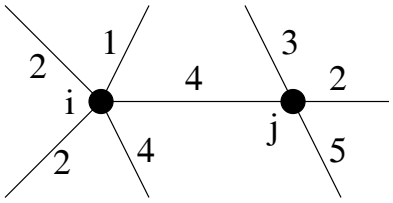
Ez az eljárás annyiban különbözik az előzőtől, hogy a lefedett gráf kiértékelésekor a minimális költségű gépek közül olyat választunk, amelynek a bemenő anyagait már gyártjuk, vagy ha ilyen nincs, akkor olyat, amelynek bemenő anyagait a legolcsóbban tudjuk előállítani. A tesztelés eredménye azt mutatja, hogy ez a heurisztika javulást eredményez a megoldások költségében.

7.5.3. A 3. heurisztika

Ez az algoritmus az előző, 2. heurisztikára épül. Az eltérés a lefedendő gráf éleinek súlyozásában van. A G_i gráfot az 1. heurisztikában leírt módon konstruáljuk meg, majd minden G_i-beli $c((p_k; p_l))$ élsúlyt helyettesítsünk $c'((p_k; p_l))$-vel, ahol:

$$c'((p_k; p_l)) = \frac{\sum_j c(p_k; p_j) + \sum_j c(p_j; p_l)}{d(p_k) + d(p_l)}.$$

Példa:



$c(i;j) = 4$,
$c'(i;j) = \frac{(1+2+2+4+4)+(3+2+5+4)}{5+4} = 3$.

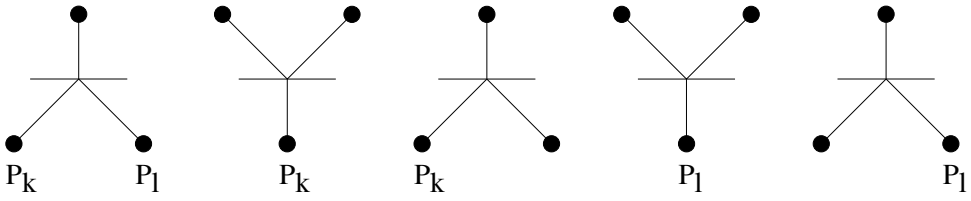
7.5.4. A 4. heurisztika

P_i-n teljes gráf G_i.

Az élek súlyozását a következőképpen végezzük:

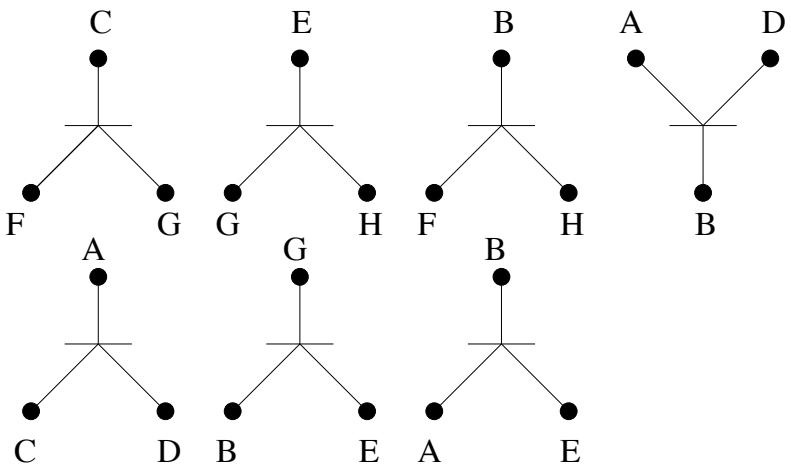
$$c((p_k;p_l)) = \min\{w(A);\ (w(B)\ vagy\ w(C))\ +\ (w(D)\ vagy\ w(E))\},$$

ahol az alábbi ábra mutatja rendre az A, B, C, D, E típusú gépeket. Az adott anyagot gyártó, adott típusú gépek közül a minimális költségű gép súlyát jelenti a w érték.



7.6. A heurisztikák működésének bemutatása

Ebben a részben egy adott példán mutatjuk be a heurisztikák működését. A példában az $\{A, B, C, D, E, F, G, H\}$ anyagok szerepelnek, nyersanyagunk nincs, az $\{F, G, H\}$ anyagokat szeretnénk előállítani. A feladat formálisan a



7.2. ábra. A rendelkezésre álló gépek

következőképpen adható meg:

$$\begin{aligned} M &= \{A, B, C, D, E, F, G, H\} \\ R &= \emptyset \\ P &= \{F, G, H\} \\ O &= \{u_1 = (C; F, G), u_2 = (E; G, H), u_3 = (B; F, H), u_4 = (A, D; B), \\ &\quad u_5 = (A; C, D), u_6 = (G; B, E), u_7 = (B; A, E)\} \\ w &= (2, 7, 3, 4, 6, 9, 4) \end{aligned}$$

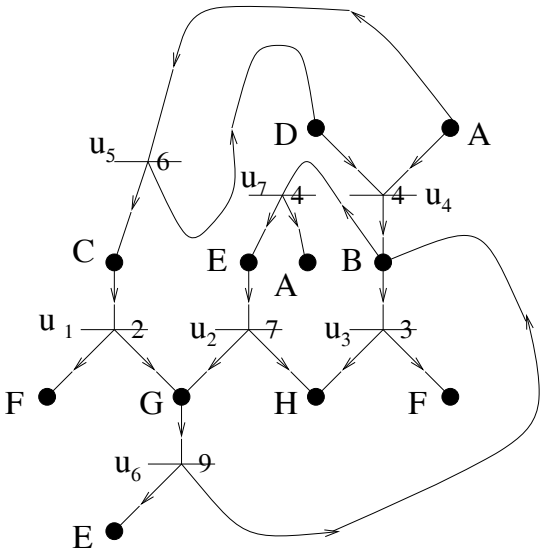
Az 7.2 ábrán láthatók a gépek. A feladathoz tartozó folyamat gráf a 7.3 ábrán látható. Az ábrán vannak anyagok, melyek több példányban szerepelnek. Így az ábra kevésbé bonyolult. A megoldás folyamata a következő:

Kiindulás: $RM_0 = P,\ PM_0 = R,\ G_0 = \emptyset,\ E_0 = \emptyset,\ O_0 = \emptyset$

1. lépés: $RM_1 = \{F, G, H\},\ PM_1 = \emptyset,\ G_1 = (F, G, H; (F, G), (F, H), (G, H))$
$E_1 = (F, G, H; (F, G), (F, H)),\ O_1 = O_0 \cup \{u_1, u_3\}.$

2. lépés: $RM_2 = \{B, C\},\ PM_2 = \{F, G, H\},\ G_2 = \emptyset, O_2 = \{u_1, u_3\} \cup \{u_4\}.$

3.lépés: $RM_3 = \{A, C, D\},\ PM_3 = \{B, F, G, H\},\ G_3 = (C, D; (C, D)),$
$E_3 = G_3,\ O_3 = \{u_1, u_3, u_4\} \cup \{u_5\}.$



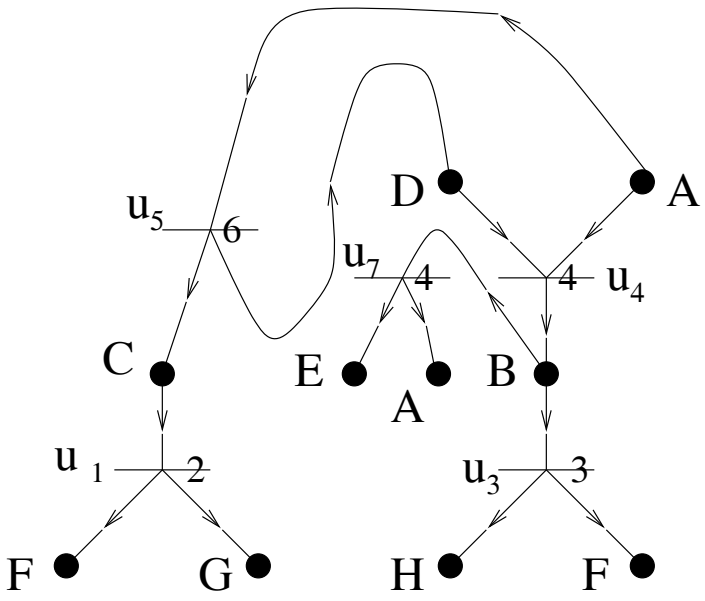
7.3. ábra. A példához tartozó folyamat gráf.

4. lépés: $RM_4 = \{A\},\ PM_4 = \{B, C, D, F, G, H\},\ G_4 = \emptyset,\ O_4 = \{u_1, u_3, u_4, u_5\} \cup \{u_7\}$.

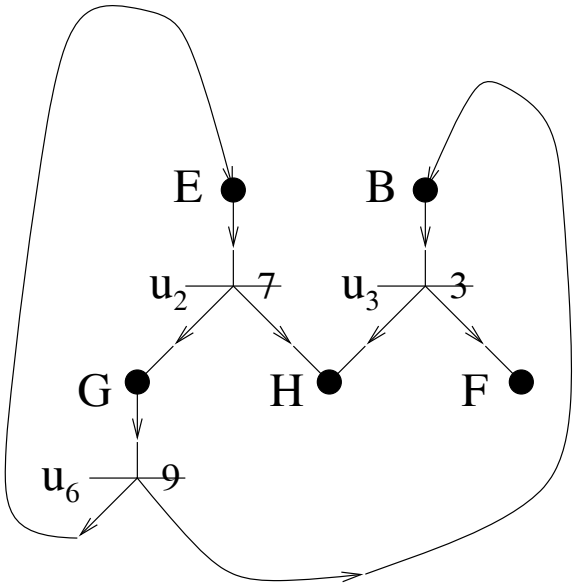
Leállás: $RM_5 = \emptyset,\ PM_5 = \{A, B, C, D, E, F, G, H\}$,
$w(O_4) = 19$
$O_4 = \{u_1, u_3, u_4, u_5, u_7\}$

Látható, hogy a megoldásban az u_1, u_3, u_4, u_5, u_7 gépek szerepelnek. A megoldás költsége: 19.

A többi heurisztika ugyanezt a megoldást adja, említést érdemel a 4. heurisztika esetében jelentkező különbség. Ebben az esetben az 1. lépés azonos lesz az 1. heurisztikánál látott 1. lépéssel, mert már ott is teljes gráfot fedtünk le. Az eltérés a 2. lépéstől jelentkezik, mert teljes gráfot kapunk itt is, így a B és C kimenő anyagokhoz tartozó gépeket már ebben a lépésben bevesszük a megoldásba. A 3. lépés azonos lesz az 1. heurisztika esetében látott 4. lépéssel. Látható, hogy a 4. heurisztika alkalmazásával kevesebb lépésben kaphatunk egy lehetséges megoldást.



7.4. ábra. A heurisztikus megoldáshoz tartozó folyamat gráf



7.5. ábra. Az optimális megoldáshoz tartozó folyamat gráf.

Egy Branch and Bound típusú eljárás által adott optimális megoldásban az u_2, u_3, u_6 gépek szerepelnek, az optimum értéke 19. Látható, hogy bár az általunk talált megoldásokban más gépek szerepelnek, ebben a példában a heurisztikák optimális megoldást találtak. A 7.4 ábrán a heurisztikák által talált megoldáshoz tartozó folyamat gráf látható, míg a 7.5 ábrán a B&B eljárással talált optimális megoldáshoz tartozó gráf. Megjegyzendő még, hogy elképzelhető olyan feladat is, melyben nyersanyagokat is gyártunk, valamint végtermékeket újra felhasználunk. A heurisztikák alkalmasak ilyen típusú feladatok megoldására is. Az is elképzelhető, hogy a megoldásban diszjunkt körök adódnak akár a nyersanyagok teljes kizárásával is, tehát a rendszer öntápláló. Ezek a megjegyzések nem idegenek a gyakorlati alkalmazásoktól, hiszen a vegyiparban található olyan gyártási folyamat, melyben végterméket táplálunk be kezdetben mint nyersanyagot, és később a gyártás során a nyersanyagot újra felhasználjuk. A kénsav gyártása is így modellezhető.

7.7. Teszteredmények PNS feladatok különféle osztályaival

Ebben a fejezetben teszteket végzünk, melyek során különféle PNS feladatokat oldunk meg a heurisztikus algoritmusok, illetve egy optimális megoldást szolgáltató Branch and Bound típusú eljárás segítségével. A lehetőségeket korlátozta, hogy bonyolultabb feladatok optimális megoldásának meghatározása túlságosan nagy időigényű a megoldás módjából adódóan. A feladatok a programhoz készült PNS probléma generáló eljárással lettek előállítva, a generált feladatok megfelelnek a rájuk vonatkozó feltételeknek. A táblázatokban szereplő feladatok optimális megoldásának meghatározása egy 466 MHz-es Celeron számítógépen percekig, egyes feladatoknál órákig tartott, míg a heurisztikák segítségével néhány másodpercen belül találtunk megoldást.

A táblázatokban szereplő jelölések:

$|O|$: az adott algoritmussal kapott megoldásban szereplő gépek száma,
$w(O)$: a megoldás költsége,
$\frac{w(O)}{|O|}$: a gépek átlagos száma az egyes algoritmusok által adott megoldásokban,
$\frac{w(O)}{w(O^{opt})}$: a heurisztikák által adott megoldások költségeinek viszonya az optimumhoz.

	B&B	Alg 1	Alg 2	Alg 3	Alg 4
megoldható	44	41	38	41	42
$\|O\|$	4,18	6,12	6,00	6,22	6,86
$w(O)$	32,75	53,44	54,29	54,54	51,98
$\frac{w(O)}{\|O\|}$	7,83	8,73	9,05	8,77	7,58
$\frac{w(O)}{w(O^{opt})}$	1,00	1,63	1,66	1,67	1,59

1. 50 feladat, $|M| = 18, |R| = 2, |P| = 3, |O| = 40, 1 \leq w(O) \leq 25$.

	B&B	Alg 1	Alg 2	Alg 3	Alg 4
megoldható	48	42	40	42	40
$\|O\|$	6,08	7,67	7,88	7,67	9,00
$w(O)$	96,56	130,4	134,7	130,4	132,77
$\frac{w(O)}{\|O\|}$	15,88	17,01	17,10	17,01	14,75
$\frac{w(O)}{w(O^{opt})}$	1,00	1,35	1,39	1,35	1,38

2. 50 feladat, $|M| = 25, |R| = 4, |P| = 5, |O| = 45, 1 \leq w(O) \leq 50$.

	B&B	Alg 1	Alg 2	Alg 3	Alg 4
megoldható	48	45	48	48	48
$\|O\|$	7,55	9,09	9,15	9,00	11,71
$w(O)$	115,49	176,24	174,73	176,8	133,44
$\frac{w(O)}{\|O\|}$	15,30	19,39	19,10	19,64	11,40
$\frac{w(O)}{w(O^{opt})}$	1,00	1,53	1,51	1,53	1,16

3. 50 feladat, $|M| = 30, |R| = 8, |P| = 10, |O| = 76, 1 \leq w(O) \leq 50$.

A táblázatokból látható, hogy általában a 4. heurisztika adja az optimumtól legkevésbé eltérő megoldást, viszont ezekben a megoldásokban van szükség a legtöbb gépre. A 3. táblázatból kiolvasható, hogy több gépet tartalmazó feladatoknál rendkívül nagy javulást érhetünk el a 4. heurisztika segítségével a többihez képest. Esetünkben csupán 16 százalékos volt az eltérés az optimumtól. Megállapítható az is, hogy általában a 3. heurisztika adja a legkevesebb gépet tartalmazó megoldást.

8. fejezet

Dominál halmazok de Bruijn gráfokban

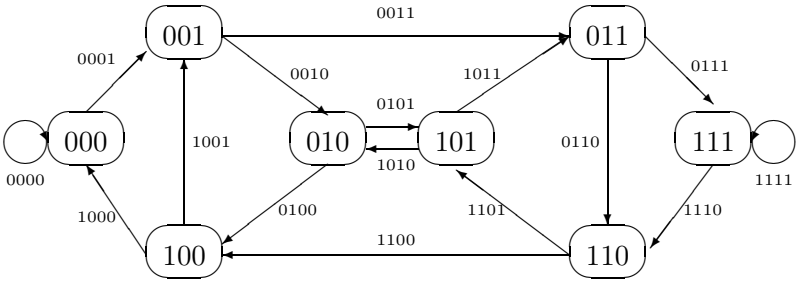
Számítógépek hálózatának építése számos nagyon érdekes matematikai, informatikai problémát vet fel. Takarékossági követelmények miatt fontos, hogy adott számú számítógép közötti direkt kapcsolatból a csúcsok számához képest kevés legyen. A kis átlagfokszám ellenére azonban legyen elég kicsi a hálózat átmérője is. A szokásos hálózati topológiák ezért nagyon erős szimmetria feltételekkel rendelkeznek. A hiperkocka egy jó példa erre, de a ***de Bruijn gráfok*** is alapvetőek, mint alternatív lehetőség. Ha arra gondolunk, hogy párhuzamosan kapcsolt gépek közül bizonyosak szerver szerepet töltenek be, míg mások csak adatbázisok, akkor fontos a szerverek optimális elhelyezkedése. Célszerű a hálózat olyan csúcsaiba telepíteni a fő gépeket, amelyekből bármely más csúcs elérhető, vagy kevés lépésben elérhető. A vizsgálatunkat a [42] cikk motiválta, az abban felvetett egyik nyitott kérdést sikerült megválaszolni. Amit korábban Biggs ***perfekt d-kódnak*** nevezett el, azt [42] nyomán itt mi is ***perfekt d-dominál halmaznak*** hívjuk. A továbbiakban dominál, hatékonyan dominál csúcshalmazokat keresünk de Bruijn gráfokban, velük kapcsolatos kérdésekre adunk további válaszokat, bizonyítunk perfekt dominál halmazokra vonatkozó pozitív és negatív eredményeket.

8.1. Alapfogalmak

Legyen $\mathcal{A}$ egy ábécé, $|\mathcal{A}| = n$. A de Bruijn gráfokat definiáljuk a következő módon:

$$B(n,k) = (V(n,k), E(n,k)),$$

ahol $V(n,k) = \mathcal{A}^k$ a gráf csúcsainak halmaza az ábécé betűiből alkotott összes k hosszú szóból áll. Az irányított élek $E(n,k) = \mathcal{A}^{k+1}$ azonosíthatók a $k+1$ hosszú szavakkal, mert az $x_1x_2\ldots x_k$ csúcsból az $y_1y_2\ldots y_k$ csúcsba pontosan akkor vezet irányított él, ha $x_2x_3\ldots x_k = y_1y_2\ldots y_{k-1}$. A 8.1 ábrán a $B(2,3)$ egy lehetséges rajza látható.



8.1. ábra. A $B(2,3)$ de Bruijn gráf.

Egy $G = (V, E)$ gráfban egy y csúcsot *dominál* egy x csúcs (másképpen x *dominálja* y-t) ha létezik egy irányított él x-től az y-ba, vagy $x = y$. A csúcsok egy $D \subseteq V$ részhalmaza *dominál halmaza* G-nek, ha a G minden csúcsát dominál legalább egy D-beli csúcs. A G legkisebb elemszámú dominál halmazának méretét hívjuk a G *dominálási számának*. A G-ben bármely ekkora dominál halmazt *minimális dominál halmaznak* nevezünk. Ha a G mindegyik csúcsát a D pontosan egy csúcsa dominál, akkor a D-t a G *perfekt dominál halmazának (PDS)* nevezzük. Egy x csúcs *d-dominál* egy y csúcsot, ha létezik G-ben legfeljebb d hosszú irányított út x-től y-ig. A csúcsok egy $D \subseteq V$ részhalmaza *d-dominál halmaza* G-nek, ha a G minden csúcsát d-dominál legalább egy D-beli csúcs. Egy ilyen D halmaz *perfekt d-dominál halmaz (d-PDS)*, ha a G mindegyik csúcsát csak egyetlen D-beli pont d-dominál.

8.2. Minimális dominál halmazok

8.2.1. Tétel. *A $B(2,k)$ de Bruijn gráfban egy minimális dominál halmaz $\left\lceil \frac{2^k}{3} \right\rceil$ csúcsot tartalmaz.*

Bizonyítás A $B(2,k)$ minden pontjának kifoka kettő, tehát egy csúcs legfeljebb két másikat tud dominálni (létezik hurokél is). Emiatt bármely dominál halmaznak legalább $\lceil 2^k/3 \rceil$ eleme van. Megadunk egy dominál halmazt, amely éppen ennyi csúcsból áll. Tekintsük az $\mathcal{A}$ ábécé elemeit a 0 és 1 számjegyeknek. Ekkor a $B(2,k)$ csúcsai a k hosszú bináris szavak. Emiatt a továbbiakban a csúcsokat természetes módon azonosíthatjuk a következő nemnegatív számokkal: 0, 1, 2, $\ldots 2^k - 1$, mint a bináris szavak értékeivel. Például a $k = 3$ esetben létezik három elemű dominál halmaz. A 4 csúcs önmagán kívül dominálja a 0 és az 1 csúcsot, az 5 dominálja még a 2-t és a 3-at, végül a 7 pedig a 6-ot. A fenti csúcshalmazt a következő eljárással kaptuk:

a 4 által dominálta	0, 1
az 5 által dominálta	2, 3
a 6 által dominálta	4, 5 – már dominálta, így a 6-ra nincs szükség
a 7 által dominált a	6

Az általános esetben így konstruálható meg egy minimális dominál halmaz:

2^{k-1}	által dominált	0 és 1
$2^{k-1}+1$	által dominált	2 és 3
$\ldots$		
$2^{k-1}+2^{k-2}-1$	által dominált	$2^{k-1}-2$ és $2^{k-1}-1$
2^{k-3}	következő számra nincs szükség	
$2^{k-1}+2^{k-2}+2^{k-3}$	által dominált	$2^{k-1}+2^{k-2}$ és $2^{k-1}+2^{k-2}+1$
$\ldots$		
$2^{k-1}+2^{k-2}+2^{k-3}+2^{k-4}-1$	által dominált	$2^{k-1}+2^{k-2}+2^{k-3}$ és $2^{k-1}+2^{k-2}+2^{k-3}+1$

2^{k-5}	következő számra nincs szükség
...	
$2^k - 1$	által dominált a $2^k - 2$ ha k páratlan, egyébként semmi más.

A fenti eljárás egy $2^k - (2^{k-1} + 2^{k-3} + 2^{k-5} + \ldots + 2^{k-\lceil k/2 \rceil + 1})$ elemű dominált halmazt ad. Indukcióval könnyű belátni, hogy ez az összeg éppen $\lceil 2^k/3 \rceil$. □

Az előző tétel természetes általánosítása a következő tétel.

8.2.2. Tétel. *A $B(n,k)$ de Bruijn gráfban egy minimális dominált halmaz*

$$\left\lceil \frac{n^k}{n+1} \right\rceil$$

csúcsból áll.

8.3. Perfekt d-dominált halmazok

M. Livingston és Q. F. Stout igazolta [42] a következő eredményt (Theorem 2.12).

8.3.1. Tétel. *Tetszőleges $d \geq 1$ esetén ha a k pozitív egész $(d+1)m$ vagy $(d+1)m - 1$ alakú, illetve $k < d$, jelölje T_k a $B(2,k)$ csúcsainak következő részhalmazát:*

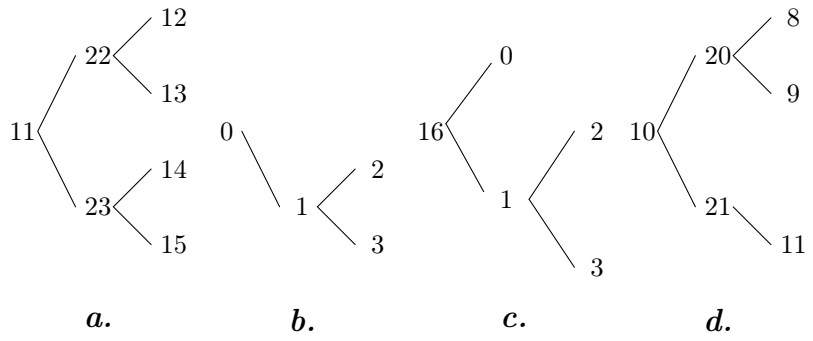
(i) $T_1 = T_2 = \ldots = T_d = \{0\}$,

(ii) $T_{(d+1)(m+1)-1} = T_{(d+1)m-1} \cup \{j : 2^{(d+1)m-1} \leq j \leq 2^{(d+1)m} - 1\}$,

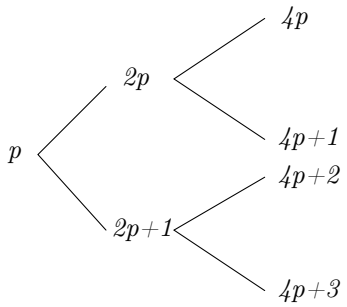
(iii) $T_{(d+1)m} = T_{(d+1)m-1} \cup \{2^{(d+1))m} - 1 - s : s \in T_{(d+1)m-1}\}$.

Ekkor a T_k perfekt d-dominált halmaza $B(2,k)$-nek.

A [42] cikkben megfogalmazták a következő sejtést: *Talán nincsenek is a fent definiálttól különböző perfekt d-dominált halmazok. Speciálisan $B(2,k)$-ban talán egyáltalán nem is található 2-dominált halmaz, amikor $k-1$ a 3 többszöröse.* A következő tétel állítja, hogy az utóbbi sejtés igaz.



8.2. ábra. Különböző típusú csúcshalmazok a 2-dominálás kor a de Bruijn gráfban ($k = 5$)



8.3. ábra. A legáltalánosabb egy csúcs által 2-dominált "sziget" a de Bruijn gráfban. Az értékek mod 2^k értendőek.

8.3.2. Tétel. *A $B(2, k)$ de Bruijn gráfban nincs perfekt 2-dominál halmaz, ha $k - 1$ a 3 többszöröse.*

Bizonyítás Jelölje $N_d(G, v)$ a $G = (V, E)$ gráf azon csúcsainak halmazát, amelyek legfeljebb d lépésben elérhetők v-ből (irányított gráfban irányított értelemben). Ha a D egy perfekt d-dominál halmaza G-nek, akkor az $\{N_d(G, v) \mid v \in D\}$ csúcshalmazok (szigetek) a V egy partícióját adják. Jelölje $n_d(G, v) = |N_d(G, v)|$ a v szigetének méretét. Ekkor $\sum_{v \in D} n_d(G, v) = |V|$ teljesül.

1. tulajdonság: $n_2(B(2, k), v) = 4, 5, 6$ or 7.

$n_2(B(2, k)), v)$ csak a következő esetekben tér el 7-től (l. a 8.2. ábrát).
$p = 2p \Rightarrow p = 0 \Rightarrow n_2(B(2, k), 0) = 4$

$p + 2^k = 2p + 1 \Rightarrow p = 2^k - 1 \Rightarrow n_2(B(2,k), 2^k - 1) = 4$
$2p + 2^k = 4p \Rightarrow p = 2^{k-1} \Rightarrow n_2(B(2,k), 2^{k-1}) = 5$
$2p + 2^k = 4p + 2 \Rightarrow p = 2^{k-1} - 1 \Rightarrow n_2(B(2,k), 2^{k-1} - 1) = 5$

Ha a k páratlan:

$p + 2^k = 4p + 2 \Rightarrow p = \frac{2^k-2}{3} \Rightarrow n_2(B(2,k), \frac{2^k-2}{3}) = 6$
$p + 2^{k+1} = 4p + 1 \Rightarrow p = \frac{2^{k+1}-1}{3} \Rightarrow n_2(B(2,k), \frac{2^{k+1}-1}{3}) = 6$

Ha a k páros:

$p + 2^k = 4p + 1 \Rightarrow p = \frac{2^k-1}{3} \Rightarrow n_2(B(2,k), \frac{2^k-1}{3}) = 6$
$p + 2^{k+1} = 4p + 2 \Rightarrow p = \frac{2^{k+1}-2}{3} \Rightarrow n_2(B(2,k), \frac{2^{k+1}-2}{3}) = 6$

2. tulajdonság: Ha $B(2,k)$-ban a p csúcs d-dominálja a q csúcsot, és a q páros, akkor a p d-dominálja a $(q+1)$-et is. Ha a p d-dominálja a q-t, és q páratlan, akkor a p d-dominálja $(q-1)$-et is.

3. tulajdonság: Ha $n_2(B(2,k), v) = 5$, akkor v nem lehet egy 2-PDS eleme.

Konkrét esetvizsgálattal ellenőrizhető. $n_2(B(2,k), v) = 5$ csak két csúcsra teljesül, $v = 2^{k-1}$ és $v = 2^{k-1} - 1$.-re.
a) Ha 2^{k-1} egy 2-PDS eleme lenne, akkor $2^{k-1} + 1$ nem lehet benne, hiszen mindketten 2-dominálják a 2-t. De ha meg a $2^{k-1} + 1$ nem lenne a 2-PDS eleme, akkor vele együtt 2-dominálva lenne a 2^{k-1} is mégegyszer. Tehát a $v = 2^{k-1}$ nem lehet egy 2-PDS-ben.
b) Ha a $2^{k-1} - 1$ lenne egy 2-PDS eleme, akkor a $2^{k-1} - 2$ nem lehet benne, hiszen ekkor mindketten 2-dominálnák a $2^k - 4$-et. Ha pedig a $2^{k-1} - 2$ nincs a 2-PDS-ben, akkor vele együtt egy csúcs 2-dominálja a $2^{k-1} - 1$-et is, de ő már benne van a a 2-PDS-ben, nem 2-dominálhatja még egy csúcs. Így tehát nem lehet 5 méretű sziget a csúcspartícióban.

4. tulajdonság: Egy 2-PDS-ben legfeljebb egy v csúcsra teljesül az $n_2(B(2,k), v) = 6$ egyenlőség.

Például a $\frac{2^k-1}{3}$ és $\frac{2^{k+1}-2}{3}$ dominálják egymást.

Tegyük most fel, hogy $k = 3l+1$ és $l > 0$. Ha az egyes típusú csúcsok számát a 2-PDS -ben A, B és C jelöli, akkor fennáll a következő összefüggés:

$$4A + 6B + 7C = 2^{3l+1} = 2((2^3)^l - 1) + 2$$

valamint a fenti tulajdonságok alapján tudjuk, hogy az A értéke 0, 1 vagy 2 és a B pedig 0 vagy 1. Ha egy mátrixban kiszámítjuk a $4A+6B-2$ kifejezés lehetséges értékeit, akkor azt tapasztaljuk, hogy egyik sem lesz osztható 7-tel. Ebből arra a következtetésre jutunk, hogy nincs 2-PDS a $B(2, 3l+1)$ de Bruijn gráfban.

8.4. Az irányítatlan de Bruijn gráfok esete

M. Livingston és Q. F. Stout a [42] cikkben az irányítatlan esettel is foglalkozik. Ha elhagyjuk az élek irányítását a $B(n, k)$-ban, akkor megkapjuk az irányítatlan de Bruijn gráfot. Jelölje tehát $B^*(n, k)$ az irányítatlan de Bruijn gráfot. Most tehát az $x_1x_2 \ldots x_k$ és $y_1y_2 \ldots y_k$ csúcsok pontosan akkor vannak összekötve, ha $x_2x_3 \ldots x_k = y_1y_2 \ldots y_{k-1}$ vagy $x_1x_2 \ldots x_{k-1} = y_2y_3 \ldots y_k$ teljesül. Hasonlóan adható meg a *dominálás* (és *d-dominálás*) fogalma is. A [42] munkában olvasható a tény, hogy $B^*(2, k)$ rendelkezik perfekt dominál halmazzal (PDS) k=1 vagy 2 esetén, de nincs PDS k=3, 4 vagy 5-re. Belátjuk, hogy $B^*(2, k)$ csak k=1 vagy k=2-re rendelkezik PDS-sel.

5. tulajdonság: $n_1(B^*(2, k), v) = 3, 4$ or 5.

Tekintsük egy tetszőleges csúcsát $B^*(2, k)$-nek. Továbbra is tekinthetjük mint egy természetes szám bináris reprezentációját. Így $N_1(B^*(2, k), p) = \left\{p, 2p, 2p+1, \lfloor p/2 \rfloor, 2^{k-1} + \lfloor p/2 \rfloor\right\}$ ahol minden természetes számot mod 2^k vegyünk. A 0 és a $2^k - 1$ csak három-három csúccsal szomszédos (és ebbe már önmagát is beleértettük). Pl. a 0 dominálja a 0, 1-et és a 2^{k-1}-et. Két olyan csúcs van, amelynek a "szigete"–az általa dominált csúcsok–négyelemű: $n_1(B^*(2, k), 2^{k-2}+2^{k-4}+\ldots+2) = 4$ és $n_1(B^*(2, k), 2^{k-1}+2^{k-3}+\ldots+1) = 4$ ha k páratlan és $n_1(B^*(2, k), 2^{k-2} + 2^{k-4} + \ldots + 1) = 4$ és $n_1(B^*(2, k), 2^{k-1} + 2^{k-3} + \ldots + 2) = 4$ ha k páros. Tehát másképpen mondva $B^*(2, k)$-ben van két hurokél, és két párhuzamos él az alternáló 0-1 szavak között (l. a 8.1. ábrát, most tekintsünk el az irányítástól). Egyébként bármely más csúcsra $n_1(B^*(2, k), v) = 5$.

6. tulajdonság: Egy PDS legfeljebb egy 4 méretű szigetet tartalmaz.

Ez az állítás nyilvánvaló a gráf szerkezete alapján.

8.4.1. Tétel. *$k > 2$ esetén nincs PDS $B^*(2,k)$-ban.*

Bizonyítás Tegyük fel, hogy mégis lenne egy D PDS $B^*(2,k)$-ban. Különböztessünk meg négy esetet a k négyes maradéka alapján. A fenti tulajdonságokat figyelembe véve mind a négy esetben csak egyetlen lehetőség marad a szigetek méretére. Ezen méretek összege megadja a csúcsok számát:

a) $3+3+5t = 2^k$ ha $k = 4l$ alakú,

b) $3+4+5t = 2^k$ ha $k = 4l+1$ alakú,

c) $4+5t = 2^k$ ha $k = 4l+2$ alakú,

d) $3+5t = 2^k$ ha $k = 4l+3$ alakú.

Az *a)* esetben nincs 4 méretű sziget. Tehát a két alternáló szó nincs a D-ben, $2^{k-2}+2^{k-4}+\ldots+2^2+1 \notin D$ és $2^{k-1}+2^{k-3}+\ldots+2 \notin D$. Ahhoz, hogy ezeket a szavakat is dominájuk, két lehetőségünk van. Az egyik, hogy $2^{k-3}+2^{k-5}+\ldots+2 \in D$ és $2^{k-1}+2^{k-2}+2^{k-4}+\ldots+2^2+1 \in D$. A másik eset hasonló, csak a 0 és 1 jegyeket kell felcserélni a bináris alakjukban. Elegendő csak az egyiket esetet részleteznünk. Vegyük a $2^{k-3}+2^{k-5}+\ldots+2+1$ szót! Ezt szintén nem tehetjük bele a D-be, mert $2^{k-3}+2^{k-5}+\ldots+2$ dominálja $2^{k-4}+2^{k-6}+\ldots+1$-t és $2^{k-3}+2^{k-5}+\ldots+2+1$ szintén dominálja őt. De nem tehetjük D-be sem a $2^{k-4}+2^{k-6}+\ldots+1$, sem a $2^{k-1}+2^{k-4}+2^{k-6}+\ldots+1$ szót, mert ezeket dominálja a $2^{k-1}+2^{k-3}+\ldots+2$ szó. Azonban a másik két szomszédja a $2^{k-3}+2^{k-5}+\ldots+2+1$ szónak a $2^{k-2}+2^{k-4}+\ldots+2^2+2$ és a $2^{k-2}+2^{k-4}+\ldots+2^2+2+1$. Viszont mindkét szó dominálja a $2^{k-1}+2^{k-3}+\ldots+2+1$ szót, pedig ez a szó már a $2^{k-1}+2^{k-2}+2^{k-4}+\ldots+2^2+1$ szó által is dominált. Ez ellentmondás, vagyis nem lehet PDS $B^*(2,4l)$-ben, az *a)* esetnek megfelelő k-ra.

Ha $k = 4l+1$, akkor szükség van egy 4 méretű szigetre a D-ben. Az egyszerűség kedvéért a *b)* esetet csak k=5-re részletezzük. Legyen pl. $01010 \in D$. 01010 dominálja a $01010, 10100, 10101$ és 00101 szavakat. Hogyan fogjuk dominálni a 11010 és 01011 szavakat? Ezeket dominálná 10101

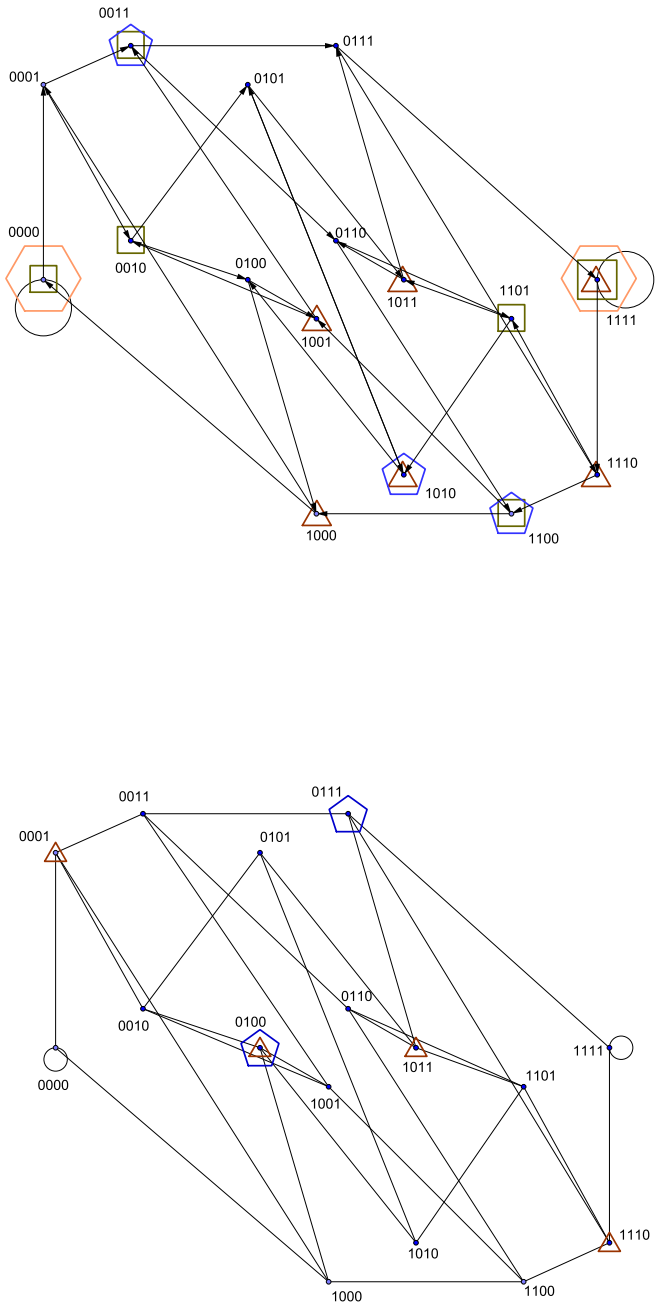
de ez a szó már dominált, így nem kerülhet D-be. Az 11010 dominálja a $11010, 01101, 11101, 10101, 10100$ halmazt, de az utóbbi kettőt már 01010 dominálja. 01011 dominálja a $01011, 00101, 10101, 10110, 10111$ halmazt, a második és harmadik szót 01010 dominálja. Így tegyük pl. a 11101 szót D-be! Most már a 10111 szót nem tehetjük D-be, mert az 11011 szót mindketten dominálnák. Ha a 10110 szóval próbálkoznánk ugyanaz lenne az ellentmondás! Ezzel a $k = 5$, esetben megmagyaráztuk, miért nincs megfelelő D és a $k = 4l + 1$ eset általában is ugyanígy megy.

A c) esetben ha volna egy megfelelő D PDS, akkor pl. $2^{k-2} + 2^{k-4} + \ldots + 1 \in D$, mert kell egy 4 méretű sziget is. $2^{k-2} + 2^{k-4} + \ldots + 1$ dominálja a $2^{k-2} + 2^{k-4} + \ldots + 1$, $2^{k-1} + 2^{k-3} + \ldots + 2$, $2^{k-3} + 2^{k-5} + \ldots + 2$, $2^{k-1} + 2^{k-3} + \ldots + 2 + 1$ halmazt. Dominálnunk kell a $2^{k-2} + 2^{k-4} + \ldots + 2^2$ és $2^{k-1}+2^{k-2}+2^{k-4}+\ldots+2^2+1$ elemeket is, hiszen ez a kettő a $2^{k-1}+2^{k-3}+\ldots+2$ még nem dominált szomszédai. De az összes szóba jöhető lehetőség rossz, ugyanis $2^{k-1} + 2^{k-3} + \ldots + 2^3 + 1$ vagy $2^{k-1} + 2^{k-3} + \ldots + 2^3$ dominálná az első szót a $2^{k-2} + 2^{k-4} + \ldots + 2^2$-t, míg $2^{k-2} + 2^{k-3} + 2^{k-5} \ldots + 2$ vagy $2^{k-1} + 2^{k-2} + 2^{k-4} + \ldots + 2$ a $2^{k-1} + 2^{k-2} + 2^{k-4} + \ldots + 2^2 + 1$ második szót, de bármelyikük ugyanazt a szót, a $2^{k-1} + 2^{k-2} + 2^{k-4} \ldots + 2^2$-t dominálja. Ez ismét ellentmondáshoz vezet.

Végül a d) eset az a)-hoz hasonlít, ennek meggondolását az olvasóra bízhatjuk.

A 8.4 ábrán felül a $B(2, 4)$ de Bruijn gráf látható (a 0 és a 15 csúcsoknál hurokéleket láthatunk). Ez a bináris ábécé feletti legkisebb olyan gráf, amelyre nincs perfekt 2-dominál csúcshalmaz (8.3.2. tétel). Négyzettel jelöltük meg az M. Livingston és Q. F. Stout által megadott kunstrukcióban a $T_4 = \{0, 2, 3, 12, 13, 15\}$ halmazt, amely egy perfekt dominál halmaz. A [13] cikkben megjelent és itt felidézett konstrukciónk által kapott minimális dominál halmazt $D = \{8, 9, 10, 11, 14, 15\}$ az ábrán háromszögek jelzik. Az ötszögekkel megjelölt {3,10,12} csúcsok egy minimális 2-dominál halmaz elemei. A $d = 3$ esetben $T_4 = \{0, 15\}$ egy perfekt 3-dominál halmaz.

Az alsó ábrán az irányítatlan $B^*(2, 4)$ de Bruijn gráf látható.
Nincs benne PDS (8.4.1. tétel)! A háromszöggel megjelölt {1,4,11,14} csúcshalmaz egy minimális dominál halmaz. Az ötszöggel jelzett csúcsok {4,7} 2-dominálják az összes csúcsot, de nem perfekt módon.



8.4. ábra. Domináló halmazok az irányított és irányítatlan de Bruijn gráfban

9. fejezet

A TSP és LOP általánosítása

Imreh Balázs szerzőtársunk, tanárunk, szeretett kollégánk 2006. augusztus 8-án elhunyt. A [14] *cikket szerzőtársaimmel az ő emlékének ajánljuk.*

Az utazó ügynök probléma inkább elméleti szempontból meghatározó jelentőségű. A gyakorlatban felmerülő problémák által felvetett optimalizálási kérdések ritkán oldhatók meg a TSP modellben. Ha az ügynöknek a valóságban meg kell látogatnia néhány várost, akkor fontos, hogy mindenhová eljusson, a végén a kiindulási helyére érkezzen, de a szokásos feltételt, hogy egyetlen kört tegyen meg az ügynök nem igazán követeli meg valamilyen takarékossági szempont.

Egy másik alapvető hálózati optimalizálási probléma az optimális sorrend (Linear Ordering Problem–LOP) megtalálásának kérdése. A csúcsok olyan sorrendjét keressük, melyre az olyan élekhez rendelt súlyok összege minimális, amely élek kezdőpontja a sorrendben megelőzi a végpontját. Tegyük fel, hogy egy postakocsi a legtöbb levelet szeretné kézbesíteni a városok között. Ezt úgy teheti meg, hogy egy útvonalon minden várost pontosan egyszer meglátogat valamilyen sorrendben, és az utolsónak érintett városban marad. Ekkor csak azokat a leveleket van módja elszállítani a címzetthez, amelyeket az útvonal korábbi állomásán adnak fel. A kézbesítésekkel bevételre tesz szert, szeretne maximális haszonhoz jutni. Vajon milyen sorrendben járja végig a városokat?

A gyakorlattól nem idegen a következő probléma. Tegyük fel, hogy

egy áruházlánc üzletei között bizonyos belső szállítási igények merülnek fel. Jobb lenne az egyik raktárkészletét egy bizonyos áruból csökkenteni, míg egy másikban éppen erre lenne szükség. A vállalat szeretne az egyik üzletéből reggel egy kamiont elindítani, amely végigjárja az áruházakat, és azokat a belső átszállításokat megoldja, amelyek esetében a feladó megelőzi az igénylőt. Minden létrejött belső szállítás haszonnal jár. Az útvonaltól függően kiadásai is vannak a kamion körútjának. Milyen útvonalat válasszon, hogy az eredő bevétel maximális legyen? Világos, hogy az utóbbi probléma a két klasszikus feladat közös általánosítása. Ebben a fejezetben ezzel foglalkozunk. Az általánosítás ötlete, az itt tárgyalt algoritmusok a dolgozat szerzőjétől származnak. A [14] cikkben további algoritmusokat definiáltunk, amelyeket Bartók Tamás megvalósított és díjazott OTDK munkát készített a futási tapasztalatok alapján.

9.1. A HPPIT probléma

Több fontos kombinatorikus optimalizálási probléma keres hálózatokban optimális költségű utakat, körutakat. Ezek közül az egyik legnépszerűbb a teljes irányított hálózatokban minimális költségű Hamilton–út keresése. Ennek körút változata a híres utazó ügynök probléma (Travelling Salesman Problem–TSP). Ez az egyik legtöbbet vizsgált optimalizálási probléma, az egyik legfontosabb NP–teljes kérdés. Változataival együtt pl. a [30] munkában olvashatunk róla. Egy másik jól ismert probléma az optimális sorrend (Linear Ordering Problem–LOP) megtalálásának feladata, ahol a célfüggvényünk az előremutató irányított élek súlyainak összege, ezt szeretnénk maximalizálni. Áttekintés olvasható a LOP probléma eredményeiről a [48] és [49] cikkekben.

Bemutatunk itt egy új modellt, amelynek a célfüggvénye a fenti két probléma célfüggvényeiből alakult ki. A motivációt a bevezetőben említett gyakorlati kérdéshez hasonlók jelentették. Tegyük fel, hogy egy jármű néhány helyet meglátogat, és a korábbiakból későbbiekbe szállíthat valamit. Ha az i.-ből a j.-be megvalósul egy szállítás, akkor ez B_{ij} hasznot hoz. Ha a cél a haszon maximalizálása a megtett út költségét leszámítva belőle, akkor kapjuk a (Min-cost Hamiltonian Path Problem with Internal Transports, rövidítve HPPIT) problémát.

A HPPIT két NP–nehéz probléma közös általánosítása, tehát maga is NP–nehéz. Divatos, de egyben fontos is nehéz optimalizálási problémákra ügyes heurisztikákat definiálni, amelyek kevés számolási igénnyel kielégítő lehetséges megoldásokat adnak. Ha azonban nem vagyunk elégedettek a heurisztikus megoldás jóságával, akkor is felhasználhatjuk a kapott lehetséges megoldásokat arra, hogy velük gyorsítsunk fel pl. egy, a Branch and Bound kereteljárásra épülő optimális megoldó eljárást. Általában a feladatok típusa és mérete együtt határozza meg, hogy mire van lehetőség, meg kell-e elégednünk a közelítő megoldással, vagy futtathatjuk a nem polinomiális optimum kereső eljárást.

Az új HPPIT modellre a TSP vagy a LOP problémákra definiált heurisztikus algoritmusokat érdemes továbbfejleszteni. Ügyelnünk kell azonban arra, hogy a HPPIT bemenő adatai, a profit mátrix és az élekhez rendelt utazási költség mátrix összemérhető értékekből álljon, ne dominájanak sem a várható haszon értékei, sem az élek költségei. Ilyen bemeneteknél várható, hogy az algoritmusaink láthatóan eltérő eredményt adjanak, ekkor érdemes a futási eredményeket egymással összehasonlítani.

9.2. Fogalmak és jelölések

Legyen G(V,A) egy irányított teljes gráf, ahol $V = \{v_0, v_1, \ldots, v_n\}$ a csúcsok halmaza (v_0 a jármű telephelye, míg a további csúcsok jelentik azokat a helyeket, amelyeket útja során meg kell hogy látogasson). Adott továbbá két $(n+1) \times (n+1)$-es nemnegatív mátrix, B és D. B_{ij} a lehetséges hasznot jelenti, ha a v_i -ből a v_j -be történő szállítás létrejön, vagyis ha v_i az útvonalon megelőzi v_j -t ($B_{ii} = 0$ minden i-re). D_{ij} adja a v_i -ből v_j -be történő közvetlen eljutás költségét ($D_{ii} = 0$ minden i-re), tehát a hálózatban az élhez rendelt súlyt.

A HPPIT problémában a telephelyről indul a jármű, minden várost pontosan egyszer érint, majd visszatér a telephelyére. Egy lehetséges megoldás tehát azonosítható az $\{1, .., n\}$ halmaz egy p permutációjával. A p által meghatározott körútban a jármű a v_0 csúcsból indul, és rendre $v_{p(1)}, v_{p(2)}, .., v_{p(n)}$ sorrendben látogatja meg a városokat. A célfüggvényt a következő formula

adja:

$$z(p) = \sum_{0<i<j<n+1} B_{p(i),p(j)} + \sum_{i=1}^{n}(B_{0,p(i)} + B_{p(i),0}) - \sum_{0\leq i<n} D_{p(i),p(i+1)} - D_{p(n),0},$$

és ezt szeretnénk maximalizálni. A telephelyről történő, és a telephelyre címzett belső szállítások teljes haszna egy konstans, független p -től.

Természetesen a megoldásról úgy is beszélhetünk mint az összes csúcson áthaladó körről. Szükség lesz egyes algoritmusoknál még meg nem épített, minden csúcsot még nem tartalmazó részkörútról beszélni. Tetszőleges részkörútra, amely tartalmazza a v_0 -t és valamely v csúcsot jelölje $PRE(v)$ a (v_0, v) út csúcsainak halmazát, és $SUC(v)$ a (v, v_0) út csúcsainak halmazát. Tehát a definiált két halmaz a részkörúttól függ, de ahol félreértésre nincs mód ezt külön nem jelöljük.

9.3. Heurisztikus algoritmusok a HPPIT problémára

A [14] cikkben bemutatott algoritmusok közül néhányat itt is bemutatunk.

9.3.1. Körútépítő technikák

KÉ1. algoritmus (Körút építő)

Egymás után definiáljuk a csúcsok sorrendjét. Minden egyes lépésben azt a csúcsot választjuk, amely a maximális nyereséget ígéri (figyelembe véve az utazási költséget is).

1. lépés Legyen $0 < k < n+1$ az az érték, ahol $\sum_{0<j<n+1} B_{kj} - D_{0k} - D_{k0} = \max_{0<i<n+1} \sum_{0<j<n+1} B_{ij} - D_{0i} - D_{i0}$. Legyen $p(1) = k$. Ha egynél több ilyen k van, akkor válasszuk a legkisebb ilyet. Legyen t=1.

2. lépés Ha $t = n$, az eljárás véget ér, definiáltuk p-t, a kapott körút $v_0, v_{p(1)}, \ldots, v_{p(n)}, v_0$. Ha $t < n$, akkor legyen $p(t)$ a legkisebb k ($k \neq p(s), s < t$) melyre: $-D_{p(t-1),k} + \sum_{0<j<n+1, j\neq p(s), s<t} B_{kj} =$

$\max_{0<i<n+1,i\neq p(s),s<t}\{-D_{p(t-1),i}+\sum_{0<j<n+1,j\neq q(s),s<t} B_{ij}\}$ Növeljük a t értékét, és folytassuk a 2. lépéssel!

KÉ2. algoritmus

Még mindig mohó módon, de most már mindkét irányba építsük a kört! Felváltva lépjünk, egyet hátulról, egyet elölről! Most is válasszunk azon lehetséges csúcsok közül, amelyek maximális haszonnal kecsegtetnek pillanatnyilag, az utazási részköltséget is számítva.

1. lépés (Az utolsó $v_{p(n)}$ és az első $v_{p(1)}$ csúcs meghatározása): Legyen $0 < k < n+1$ az az érték, melyre $\sum_{0<j<n+1} B_{jk} - D_{k0} = \max_{0<i<n+1} \sum_{0<j<n+1} B_{ji} - D_{i0}$. Ha egynél több ilyen k van, akkor válasszuk a legnagyobbat! Legyen $p(n) = k$, $F = \{k\}$ (F a már eddig rendezett csúcsok halmaza). Ha $n > 1$, akkor legyen $0 < k < n+1$, $k \notin F$ az az érték, melyre $\sum_{0<j<n+1,j\neq p(n))} B_{kj} - D_{0k} = \max_{0<i<n+1,i\notin F} \sum_{0<j<n+1,j\neq p(n)} B_{ij} - D_{0i}$. Legyen $p(1) = k$. Ha egynél több megfelelő k van, akkor válasszuk a legnagyobbat. Legyen $t = n-1$, $z = 2$, és $F = F \cup \{k\}$.

2. lépés Ebben a lépésben a hátulról következő várost $p(t)$-t választjuk ki. Ha $z = n$, akkor az eljárás befejeződik, a p permutációt megadtuk, $v_0, v_{p(1)}, \ldots, v_{p(n)}, v_0$ a körút. Ha $z < n$, akkor legyen $p(t)$ a maximális k, $k \notin F$ melyre: $-D_{k,p(t+1)} + \sum_{0<j<n+1,j\notin F} B_{jk} = \max_{0<i<n+1,i\notin F}\{-D_{i,p(t+1)} + \sum_{0<j<n+1,j\notin F} B_{ji}\}$.

Legyen $z = z+1$, $t = n-t+1$, $F = F \cup \{k\}$

3. lépés Ebben a lépésben meghatározzuk $p(t)$-t, az elölről következő indexet. Ha $z = n$, akkor az eljárás véget ér, p-t meghatároztuk, a körút a $v_0, v_{p(1)}, \ldots, v_{p(n)}, v_0$. Ha $z < n$, akkor legyen $p(t)$ a legkisebb k, $k \notin F$, melyre: $-D_{p(t-1),k} + \sum_{0<j<n+1,j\notin F} B_{kj} = \max_{0<i<n+1,i\notin F}\{-D_{p(t-1),i} + \sum_{0<j<n+1,j\notin F} B_{ij}\}$

Legyen $z = z+1$, $t = n-t$, $F = F \cup \{k\}$ és lépjünk a 2. lépésre.

KÉ3 algoritmus (A következő beillesztése)

Inicializálás Legyen $r = 0$, $I_r = 0$, $E_r = \{(0,0)\}$. (v_0 a kiindulási részkörút) Legyen $r = 0$ és lépjünk az iterációs részre!

Iterációs rész (az r. iteráció) Ha $r = n$, akkor az eljárás befejeződik, az E_r élei adják a megoldást. Egyébként válasszuk az r -hez azt az (u, v) élt az E_r halmazból, melyre

$\sum_{i \in PRE(v)} B_{ik} + \sum_{j \in SUC(u)} B_{kj} - D_{uk} - D_{kv} + D_{uv} =$
$\max_{(s,t) \in E_r} \{\sum_{i \in PRE(t)} B_{ik} + \sum_{j \in SUC(s)} B_{kj} - D_{sk} - D_{kt} + D_{st}\}$.

Legyen $E_{r+1} = E_r \setminus \{(u, v)\} \cup \{(u, r), (r, v)\}$ ahol az (u, v) a kiválasztott él. Növeljük r-et eggyel, és lépjünk a következő iterációra.

KÉ4 algoritmus (A legjobb beszúrása)

Ebben az algoritmusban minden lépésben a már meglévő részkörúthoz olyan új várost választunk, amelyet a legnagyobb nyereséggel (legkisebb költséggel) lehet beszúrni és arra a helyre illesztjük a körbe. Az algoritmus a legolcsóbb beszúrása (cheapest insertion) TSP heurisztika kiterjesztése. Az eredetit a [26] és [27] munkákban elemezték. A negyedik algoritmusunk a következő:

Inicializálás. Legyen $r = 0, I_r = 0, E_r = \{(0, 0)\}$ (v_0 a kiindulási részkörút). Menjünk az iterációkhoz!

Iterációs rész (Az r. iteráció) Ha $r = n$, akkor az eljárás befejeződik, a körút élei az E_r elemei. Egyébként határozzuk meg minden egyes k-ra az $N \setminus I_r$ halmazból a $k(u, v)$ élét az E_r halmaznak, melyre

$C(k(u, v)) = \sum_{i \in PRE(v)} B_{ik} + \sum_{j \in SUC(u)} B_{kj} - D_{uk} - D_{kv} + D_{uv} =$
$\max_{(s,t) \in E_r} \{\sum_{i \in PRE(t)} B_{ik} + \sum_{j \in SUC(s)} B_{kj} - D_{sk} - D_{kt} + D_{st}\}$.

Legyen k az az érték, ahol a fenti maximum a legnagyobb. Legyen $I_{r+1} = I_r \cup k$ és $E_{r+1} = E_r \setminus \{(u, v)\} \cup \{(u, k), (k, v)\}$ ahol (u, v) az az él, amelyre a maximális $C(k(u, v)$ érték adódott a rögzített k-ra. Növeljük r-t eggyel és lépjünk a következő iterációra.

9.3.2. Túra javító módszerek

Számos módon meg lehet próbálni javítani egy megépített körutat. Mi egy olyat alkalmaztunk, amely a szomszéd keresési algoritmusok közé tartozik. (Bővebben a [1] cikk foglalkozik a részletekkel). Mondjuk azt, egy körútnak szomszédja egy olyan körút, amelyet úgy kapunk belőle, hogy két kiválasztott csúcs helyzetét megcseréljük. Ezt a szomszédosság fogalmat az ütemezés területén is használják. A körútjavító alapeljárásunk a következő.

A körútjavító alapeljárás

Inicializálás Legyen X egy tetszőleges módon kapott körút. Pl. valamely fenti heurisztika eredménye. Legyen $X_0 = X$, $r = 0$, és folytassuk az iterációval.

Iterációs rész (r. iteráció)

1. lépés Vegyük az X_r szomszédait. If $z(X_r) \geq z(Y)$ minden egyes Y szomszédra, akkor a javítás befejeződött, nincs lokálisan jobb szomszéd. X_r a javító algoritmus eredménye.

2. lépés Legyen Y az X_r szomszédai közül egy maximális célfüggvényértékű. Legyen $X_{r+1} = Y$, az r -et növeljük eggyel, és lépjünk a következő iteráció 1. lépésére.

Természetesen a javító procedúra nem hatékony! Bár egy lépésben egy adott körút szomszédainak száma $O(n^2)$, azonban az iterációk számára nem ismert polinomiális korlát.

9.4. Számítógépes vizsgálat, értékelés

A [14] cikkben bővebben olvashatók a számítógépes vizsgálatunk eredményei. Röviden összefoglalva a definiált 6 körútépítő algoritmus viselkedését véletlenül generált mintákon teszteltük. A cél a különböző heurisztikák egymáshoz történő összehasonlítása volt. Összesen 14 algoritmust tudtunk összehasonlítani. A 6 körútépítő, ezek mindegyikének a javító procedúrával ellátott változata, valamint egy-egy algoritmus, amely minden egyes bemenetre tör-

ténő futássorozat után kiválasztja a 6 körútépítő algoritmus által adott lehetséges megoldások közül a legjobbat, illetve ezek javított változatai közül a legjobbat.

Mint a bevezetőben is említettem, nagyon fontos a véletlenül generált bemeneti mátrix párban szereplő értékek aránya, eloszlása.

- A teszt: Mindkét mátrix (B és D, $n \times n$-es) elemei a (0,500) intervallumból lettek kiválasztva, egyenletes eloszlással.
- B teszt: A B elemei továbbra is a (0,500) intervallumból lettek kiválasztva egyenletes eloszlással, míg a D mátrix elemei a (0,500n/2) intervallumból, egyenletes eloszlással.

  Az intervallumok hosszát az indokolja, hogy a célfüggvényünkben a B elemei közül $O(n^2)$ szerepel, míg a D elemei közül n.
- C teszt: Mindkét mátrix (B és D, $n \times n$-es) elemei a (400,600) intervallumból lettek kiválasztva, egyenletes eloszlással.

  Az A teszthez képest az eltolt intervallum nem engedi meg, hogy az egyes haszon értékek aránya túl nagy legyen. Ott akár 500 is lehetett, itt legfeljebb $\frac{3}{2}$.
- D teszt: A B elemei továbbra is a (400,600) intervallumból lettek kiválasztva egyenletes eloszlással, míg a D mátrix elemei a (200n,300n) intervallumból, egyenletes eloszlással.

A B és D tesztekben megpróbáltuk kiegyensúlyozni a belső szállításokból eredő haszon volumenét és az útvonal költségét. A költség n él hossza, míg a haszon $n^2/2$ érték összege. Úgy érezzük így fogható meg leginkább a HPPIT probléma egyedisége, lényeges eltérése a TSP és LOP feladatoktól. Természetesen gyakorlati problémákban a B és D mátrixban szereplő értékek teljesen függetlenek lehetnek mind egymástól, mind a többi elemtől. Az egyes mátrixokban szereplő értékek eloszlása sem feltétlenül követ szabályt. 500 mátrix párt ($n = 100$ az egyes tesztesetek szerint generálva a 9.4 Táblázatbeli célfüggvényérték átlagokat kaptuk.

	KÉ1	KÉ2	KÉ3	KÉ4
A teszt	1246680	1235690	1328640	1334750
B teszt	1086660	1092810	1087470	1111690
C teszt	2399830	2357190	2510870	2513650
D teszt	376879	379556	455414	465313

9.4 Az átlagos célfüggvényérték (KÉ alg., n=100)

Mindegyik körútépítő algoritmus nagyon gyors, az összes feladatra lefutott néhány másodperc alatt. Ha azonban a javító procedúrát is futtattuk már lényegesen nagyobb volt a futási idő. Másrészt viszont nem javítottak lényegesen a korábban kapott eredményeken.

Méginkább jelentkezett ugyanez a probléma az $n = 1000$ méretnél. Ekkor már olyan naggyá vált a futási idő a javító eljárásnak köszönhetően, hogy elértük a tesztelhetőség határát.

Az itt bemutatott körútépítő algoritmusok közül a 4. adta a legtöbb esetben a legjobb megoldást. Ez összhangban van a TSP heurisztikák közötti jósági viszonyokkal. De azért egyes feladatokra a 2., sokkal egyszerűbb algoritmus bizonyult a legjobbnak. Érdemes megjegyezni, hogy megfigyelésünk szerint a legtöbb legjobb eredményt adó 4. algoritmus jóval lassúbb mint a többi, tehát számolása nem hiábavaló. Meglepő, hogy az egyes tesztesetekben lényegesen eltérő a javító procedúra időigénye. Azonos számú és méretű feladatokra a B és D tesztekben, tehát amikor a célfüggvény két részének hozzájárulása kiegyensúlyozott, sokkal gyorsabban befejeződött a javítás. A javítás mértéke azonban az A és C teszteknél nagyobb. Ez talán abból is adódhat, hogy az igazi optimumtól távolabb van az A és C esetekben a körútépítő algoritmusokkal kapott heurisztikus megoldás értéke, vagyis többet lehet még rajta javítani.

Összefoglalás

A domináló csúcshalmazok szerepét hálózatokban csak a 8. fejezetben mutatjuk be az eredeti értelmében.

A hálózati folyamatok szintézise témában adott egy anyaghalmaz, amelyet megadott átalakítók egy része segítségével elő szeretnénk állítani. Ha műveleti egységek együtt minden kívánt anyagot előállítanak, a számukra szükséges és még nem termelt anyagokat is elő kell más gépekkel állítani. A nyersanyagokon kívül tehát a gépek összes bemeneti anyagát, és a kívánt anyagokat is dominálnia kell valamely kiválasztott gépnek. Egy géphalmaz, amely teljesíti az eddig említett feltételeket csak akkor lehetséges megoldás, ha nincs közöttük felesleges, vagyis olyan, amely nem d-dominál legalább egy kívánt anyagot valamilyen d-re.

A disszertáció harmadik témája a HPPIT probléma. A LOP probléma az egyik olyan, amelyből a HPPIT ered. Ebben, lényegét tekintve az a feladatunk, hogy megtaláljuk a csúcsoknak azt a sorrendjét, amelyre a legtöbb az előre mutató él.

Az első néhány fejezetben a PNS modellt elemezzük. A második fejezetben összefoglaljuk a PNS alapvető fogalmait és felidézünk több, korábbi eredményt. A strukturális modell, a maximális struktúra definíciója, a lehetséges megoldások legfontosabb kombinatorikus tulajdonságainak leírása megtalálható a [18], [19], [20], [21],[22], [23], [25], [34], [35] cikkekben. Az első vizsgált optimalizálási feladat az volt, amikor a műveleti egységekhez rendeltünk költséget, és minimális összköltségű lehetséges megoldást kerestünk. Erre a PNS problémára először a Korlátozás és Szétválasztás technikájával sikerült megoldást adni.

Vajon meg lehet-e oldani hatékonyan egy PNS problémát? A kombinatorikus összefüggések megértésével tudunk-e polinomiális algoritmust definiálni a problémára? A 3. fejezetben egy polinomidejű visszavezetést adunk meg (l. a [4] cikket); a PNS egy speciális osztályára vezetjük vissza az

NP–teljes halmazlefedési feladatot (3.1.1. Tétel). Ebből következik, hogy a Minsum PNS probléma NP–nehéz, tehát nem várható az általános feladatra hatékony megoldás.

A Korlátozás és Szétválasztás elvére épülő algoritmusokat mutat be a [19] cikk, amelyben bevezették a döntési leképezés fogalmát. A negyedik fejezetben tárgyaljuk az ún. konzisztens döntési leképezések számának korlátozására vonatkozó eredményeinket [5], [6], [7]. A Korlátozás és Szétválasztás elvű algoritmusok gyorsításában nagy jelentőségű a lehetséges maximális konzisztens döntési leképezések számának felülről történő jó becslése. Ha a lehetséges megoldások axiómái közül az egyiket vesszük csak figyelembe, akkor felső korlátot kaphatunk (5.1.1. Tétel). Ha a Szitaformulát szeretnénk felhasználni a korlát kiszámításához, akkor $|A_I|$ meghatározása szükséges. Ez általában nem tűnik egyszerűnek. Használható a formula szeparátor típusú műveleti egységekre, az ún. Egyenes és Lánc modelleknél sikerült pontosan kiszámolni a korlátokat, amelyek elég jóknak is bizonyultak. Az eltérő számítási módszerek hozománya két kombinatorikus azonosság is.

A célfüggvény szerint csoportosítva kombinatorikus optimalizálási feladatokat, beszélhetünk Minsum, Bottleneck vagy k-összeg változatokról. Az utóbbi kettőt először a [8] cikkben vizsgáltam PNS feladatokra. Kiderült (6.2.1 Lemma), hogy a Bottleneck feladatra adható hatékony algoritmus (6. fejezet, 1. Procedúra). A PNS k-összeg változatára bizonyítottam a 6.3.1. Tételt, az erre épülő eljárás (2. Procedúra) rögzített k-ra hatékony.

Minden P-gráfot át lehet alakítani, egy vele ekvivalens másik, egyszerűbb P-gráfra, amelynek lehetséges megoldásai alapján az eredeti feladat megoldásait visszakaphatjuk (7.1.1. Tétel). Az egyszerűsített P-gráfok csak három anyaghoz kapcsolódó gépeket tartalmaznak. Ilyen P-gráfokra sikeresen alkalmaztuk több heurisztikában az MCEC algoritmust, amely egy hálózatban minimális összsúlyú élhalmazzal fedi le a gráf csúcsait. A 7. fejezet heurisztikái minden lépésben a valódi optimumot adó MCEC algoritmust használják. Emiatt az egyes lépésekben nemcsak az időigényt tekintve működnek hatékonyan az algoritmusok. A [11] és [12] cikkekben publikáltam a módszert és elemeztük a számítógépes tapasztalatainkat nagy, generált feladatokon.

M. Livingston és Q. F. Stout a [42] cikkükben számos fontos gráfosztályra vizsgáltak dominálási kérdéseket. A de Bruijn gráfokra kimondott tételük konstrukciót adott bizonyos paraméterosztályok esetén, de nyitva hagyott további sejtéseket. A 8. fejezetben két tételben (8.3.2. és 8.4.1.) igazoltuk, hogy nem létezhetnek perfekt domináló halmazok végtelen sok de Bruijn gráfban [13]. Minimális domináló halmazra általános konstrukciót ad-

tunk.

A disszertáció nagyobb részében algoritmusokat próbáltunk gyorsítani, heurisztikákat definiáltunk és elemeztünk, különböző célfüggvényekre bizonyítottunk bonyolultsági eredményeket. A 9. fejezetben bevezettünk egy új, a valós élet által motivált kombinatorikus optimalizálási problémát, amely két híres NP–teljes probléma közös általánosítása [14]. Összesen 14 algoritmus változatot hasonlítottunk egymáshoz. Úgy gondolom egy körút során a belső szállítások által elérhető teljes haszon maximalizálása számos gyakorlati problémában megjelenik, emiatt a HPPIT problémát mind elméleti, mind gyakorlati szempontból sokan fogják még vizsgálni.

Summary

The role of dominating sets in topic of process networks presents in the original sense only in chapter 8. In Process Network Synthesis a desired material set P is given, and our goal is to find an operating unit set to produce P. A subset of operating units from a feasible solution dominate a set of materials. For every unit we have a material set as input set of the unit, which are need to be a subset of the dominated set of all units union raw material set. In a manufacturing system or in other practical application of the PNS model we need not a unit if it is not d-dominate some material from P, for a positive integer d. The third topic of the thesis is the HPPIT problem. One of the two parent problems is the LOP. We need to count all the arcs forward in the case of constant weight function. The question is, how many the sum of the number of dominated vertices consistent to the order, and which ordering give a maximal value.

In the first part of this work we study the PNS model. The second chapter is a survey of basic notions and notations of PNS. The definition of the structural model, the first combinatorial properties of the feasible solution processes, the notion of maximal structure were in [22], [23], [18], [19], [20], [21] [25], [34], [35] papers. The first goal was to find an appropriate feasible solution with minimal sum of costs of units in it. This objective function yields the so-called minsum version of the weighted PNS problems. For the solution of this PNS problem, more algorithms were developed, most of them are based on Branch and Bound technique.

How can we solve a PNS problem efficiently? Can we use combinatorial ideas for an algorithm to solve PNS in polynomial time? In chapter 3 we were able to show a nice polynomial transformation of a special case of Minsum PNS problem (2.2) to the set covering problem [4]. We proved this fundamental question of theory of PNS, the problem is NP–hard (Th. 3.1.1)!

Exponential time Branch and Bound algorithms were studied for PNS problem in [19]. In chapter 4 we considered the bounding problem of the number of consistent decision-mappings belonging to an $\mathbf{M} = (P, R, O)$ structural model. It is important to improve bounding function, and to put smaller the space of maximal consistent decision-mappings of a Branch and Bound technique. The optimization on the set of maximal consistent decision-mappings it is easier in a bounded solution space. Taking into account axiom ($\mathcal{A}2$) only, this bound can be improved (Theorem. 5.1.1). Counting of $|A_I|$ from our formula is a difficult combinatorial question in a general index set of operating units of an arbitrary P-graph. In a special PNS class we have given a complicated formula for $|A_I|$, see [5]. We considered two more special models, the so-called Line model and Chain model. The determination of $|A_I|$ was easier to these nice models. We could count a formula for these bounds with Inclusion–Exclusion Formula and with direct way, too. So we obtained two nice combinatorial identity from these counting.

The general Minmax or Bottleneck optimization problem and the k-sum versions are NP–complete problems. The Minsum version of the PNS problem is NP–complete, what we can tell about complexity of these two versions of PNS? In [8] we answered these questions. We have Procedure 1. based on Lemma (6.2.1) which solve the Bottleneck optimization problem for PNS efficiently. For the k-sum version of PNS we proved Theorem 6.3.1 and gave an algorithm to solve this problem, too. Our method for solving of k-sum version of PNS problem is polynomial for „small" fixed k.

Every P-graph can be transformed into a simplified form. This simplified form consist of simple operating units in which the total number of input and output materials is at most 3. This observation facilitates useful application of the Edge Covering Problem of weighted graphs, which can be used to develop a new heuristic procedure for the PNS problem. In chapter 7 we prove Theorem 7.1.1 about an equivalent transformation of P-graph. Using the algorithm MCEC we have defined 4 algorithms for simplified PNS problems, and gave an analysis of our computational experiments see [11] and [12]. Our main tool for the exact optimization at every step of heuristics was the algorithm MCEC, which is a blossom-type algorithm, it is very similar to the famous Edmonds' matching algorithm.

In [42] M. Livingston and Q. F. Stout gave a construction of a PDS for de Bruijn graphs in infinitely many cases, but their characterization was

not complete. We have proved two theorems, 8.3.2 and 8.4.1 about their conjectures. These results claim for infinite k parameter values that some directed and undirected de Bruijn graphs with parameter k if there have a 2-PDS or PDS, [13]. We have a construction for a minimal dominating set in directed de Bruijn graphs in general.

Most of our results try to improve algorithms, investigate simplifications for efficient heuristics, define other objective functions for a solution in polynomial time. In chapter 9 we consider a new combinatorial optimization model as a common generalization of TSP and LOP problems. These two problem are well-known NP–complete problems. Why I define HPPIT, a more complex problem with a mixed linear cost function from the parents problems? The motivation was given by practical optimization questions. The objective was to maximize the total profit achieved by the inner transportation taking into account the cost of the tour of a vehicle.

Publications of the results

Blázsik, Z., B. Imreh, A note on connection between PNS and set covering problems, *Acta Cybernetica*, **12**, 1996, 309-312. (MR1428741)

Blázsik, Z., Cs. Holló, B. Imreh, On Decision-Mappings Related to Process Network Synthesis Problem, *Acta Cybernetica*, **13**, 1998, 319-328. (MR1644388)

Blázsik, Z., Cs. Holló, B. Imreh, Explicit bound for the number of feasible solutions of special PNS-problem classes, *Pure Mathematics and Applications*, **9**, 1998, 17-27. (MR1677229)

Blázsik, Z., Cs. Holló, B. Imreh, Cs. Imreh, Z. Kovács, On Bottleneck and k-sum version of the Process Network Synthesis Problem, *Novi Sad Journal of Mathematics*, **3**, 2000, 11-19. (MR1776440)

Blázsik, Z., K. Keserű, Z. Kovács, Heuristics for simplified Process Network Synthesis problems with a Blossom-type Algorithm for the edge covering problem, Optimization Theory: Recent Developments from Matrahaza,(eds.: F. Gianessi, P. Pardalos, T. Rapcsák), Kluwer Academic Publishers, Dordrecht, 2001, 19-31. (MR1886425)

Blázsik, Z., K. Keserű, Z. Kovács, Heuristics for PNS problems and its empirical analysis, *Pure Mathematics and Applications*, **11**, 2001, 139-151. (MR1839923)

Blázsik, Z., Z. Kása, Dominating sets in de Bruijn graphs. Algebraic systems (Felix-Oradea, 2001). *Pure Mathematics and Applications*, **13** 2002, 79-85. (MR1987200)

Blázsik Z., T. Bartók, B. Imreh, Cs. Imreh, Z. Kovács, Heuristics on a Common Generalization of TSP and LOP, accepted for publication.

Tárgymutató

Irodalomjegyzék

[1] Aarts, E., J. K. Lenstra (eds) *Local Search in Combinatorial Optimization*, Wiley Interscience, Chichester, England, 1997,

[2] Aho A.V., J. E. Hopcroft, J. D. Ullman, The Design and Analysis of Computer Algorithms, *Addison-Wesley, Reading Mass*, 1974.

[3] Blázsik, Z., M. Hujter, A. Pluhár, Zs. Tuza, Graphs with no induced C_4 and $2K_2$, *Discrete Mathematics*, **115**, 1993, 51-55.

[4] Blázsik, Z., B. Imreh, A note on connection between PNS and set covering problems, *Acta Cybernetica*, **12**, 1996, 309-312.

[5] Blázsik, Z., Cs. Holló, B. Imreh, On Decision-Mappings Related to Process Network Synthesis Problem, *Acta Cybernetica* **13**, 1998, 319-328.

[6] Blázsik, Z., Cs. Holló, B. Imreh, Explicit bound for the number of feasible solutions of special PNS-problem classes, *Pure Mathematics and Applications*, **9**, 1998, 17-27.

[7] Blázsik, Z., Cs. Holló, B. Imreh, Kiszámolható korlátok speciális PNS-problémaosztályok lehetséges megoldásai számára, *Új utak a magyar operációkutatásban, szerk.: Komlósi S., Szántai T., Dialóg Campus Kiadó, Budapest-Pécs,* 1999, 182-194.

[8] Blázsik, Z., Cs. Holló, B. Imreh, Cs. Imreh, Z. Kovács, On Bottleneck and k-sum version of the Process Network Synthesis Problem, *Novi Sad Journal of Mathematics* **3**, 2000, 11-19.

[9] Blázsik, Z., Cs. Holló, B. Imreh, Cs. Imreh, Z. Kovács, On a well-solvable class of the PNS problem, *Novi Sad Journal of Mathematics*, **3**, 2000, 21-30.

[10] Blázsik, Z., Cs. Holló, Cs. Imreh, Z. Kovács, Heuristics for the Process Network Synthesis Problem, *New Trends in Equilibrium Systems, Mátraháza Optimization Days*, Kluwer Academic Publishers, 2000, 1-16.

[11] Blázsik, Z., K. Keserű, Z. Kovács, Heuristics for simplified Process Network Synthesis problems with a Blossom-type Algorithm for the edge covering problem, Optimization Theory: Recent Developments from Matrahaza,(eds.: F. Giannessi, P. Pardalos, T. Rapcsák), Kluwer Academic Publishers, Dordrecht, 2001, 19-31.

[12] Blázsik, Z., K. Keserű, Z. Kovács, Heuristics for PNS problems and its empirical analysis, *Pure Mathematics and Applications*, **11**, 2001, 139-151.

[13] Blázsik, Z., Z. Kása, Dominating sets in de Bruijn graphs. Algebraic systems (Felix-Oradea, 2001). *Pure Mathematics and Applications*, **13**, 2002, 79-85.

[14] Blázsik Z., T. Bartók, B. Imreh, Cs. Imreh, Z. Kovács, Heuristics on a Common Generalization of TSP and LOP, közlésre elfogadva.

[15] Blázsik Z., B. Imreh, Cs. Imreh, Z. Kovács, On a Bin Packing Approach of a Shipment Construction Problem, Proceedings of microCAD, 2006, 15-19.

[16] Blázsik Z., B. Imreh, Cs. Imreh, Z. Kovács, The TSP problem with internal transports, Proceedings of microCAD 2006, 9-13.

[17] Floudas, C. A., I. E. Grossmann, Algorithmic Approaches to Process Synthesis: Logic and Global Optimization, AIChE Symposium Series No. 304, **91** (Eds.: L. T. Biegler and M. F. Doherly), 1995, 198-221.

[18] Friedler, F., K. Tarján, Y. W. Huang, L. T. Fan, Graph-Theoretic Approach to Process Synthesis: Polynomial Algorithm for maximal structure generation, *Computer chem. Engng.* **17**, 1993, 924-942.

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

[20] Friedler, F., J.B. Varga, E. Fehér, L.T. Fan, Combinatorially Accerelated Branch-and-Bound Method for Solving the MIP Model of Process Network Synthesis, Int. Conf. on State of the Art in Global Optimization: Computational Methods and Applications, Princeton, 1995.

[21] Friedler, F., J. B. Varga, E. Fehér, L. T. Fan, Combinatorially Accelerated Branch-and-Bound Method for Solving the MIP Model of Process Network Synthesis, *Nonconvex Optimization and its Applications*, (eds.: C. A. Floudas and P. M. Pardalos), Kluwer Academic Publishers, Norwell, 1996, 609-626.

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

[23] Friedler, F., K. Tarján, Y. W. Huang, L. T. Fan, Combinatorial Algorithms for Process Synthesis, *Computer chem. Engng.*, **16**, 1992, 313-320.

[24] Friedler, F., J. Fülöp, B. Imreh, On the reformulation of some classes of PNS-problems as set covering problems, *Acta Cybernetica*, **13**, 1998, 329-337.

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

[26] Friese A. M., G. Galbiati, F. Maffioli, On the worst-case performance of some algorithms for the asymmetric traveling salesman problem, *Networks*, **12**, 1982, 23-39.

[27] Golden, B., L. Bodin, T. Doyle, W. Stewart Jr., Approximate traveling salesman algorithms, *Operations Research*, **28**, 1980, 694-771.

[28] Grossmann, I. E., V. T. Voudouris, O. Ghattas, Mixed-Integer Linear Programming Reformulations for Some Nonlinear Discrete Design Optimization Problems, In: Recent Advances in Global Optimization (Eds.:

C. A. Floudas and P. M. Pardalos) Princeton University Press, New Jersey, 1992.

[29] Gupta, S. K., A. P. Punnen, k-sum optimization problems, *Oper. Res. Letters*, **9**, 1990, 121-126.

[30] Gutin, G., A. P. Punnen, ed., The traveling salesman problem and its variations, Kluwer Academic Publisher, Dordrecht, 2002

[31] Holló, Cs., Z. Blázsik, Cs. Imreh, Z. Kovács, On a Merging Reduction of the Process Network Synthesis Problem, *Acta Cybernetica*, **14**, 1999, 251-261.

[32] Holló, Cs. Hálózati folyamatok szintézise, Doktori diszertáció, Szeged, 2004

[33] Holló, Cs., A Look Ahead Branch-and-Bound Procedure for Solving PNS Problems, *Pure Mathematics and Applications*, **2**, 2000, 265-279.

[34] Imreh, B., F. Friedler, L. T. Fan, An Algorithm for Improving the Bounding Procedure in Solving Process Network Synthesis by a Branch-and-Bound Method, *Developments in Global Optimization*, ed. I. M. Bomze, T. Csendes, R. Horst, P. M. Pardalos, Kluwer Academic Publisher, Dordrecht, 1996, 301-348.

[35] Imreh, B., G. Magyar, Empirical Analysis of Some Procedures for Solving Process Network Synthesis Problem, *Journal of Computing and Information Technology*, **6**, 1998, 373-382.

[36] Imreh, B., *Kombinatorikus optimalizálás*, Novadat Győr, 2000.

[37] Imreh, B., J. Fülöp, F. Friedler, A Note on the Equivalence of the Process Network Synthesis and Set Covering problems, *Acta Cybernetica*, **14**, 2000, 497-502.

[38] Imreh, Cs., Jól megoldható PNS osztályokról, *Új utak a magyar operációkutatásban, szerk.: Komlósi S., Szántai T., Dialóg Campus Kiadó, Budapest-Pécs*, 1999, 168-181.

[39] Imreh, Cs., A new well-solvable class of PNS problems, *Computing*, **66**, 2001, 289-296.

[40] Imreh, Cs., Combinatorial algorithms for the PNS and online scheduling problems, Doctoral thesis, Szeged, 2001.

[41] Karp, R. M., Reducibility among Combinatorial Problems in Complexity of Computer Computations, *R. E. Miller and T. W. Thatcher, eds.*, Plenum Press, New York, 1972.

[42] Livingston, M., Q. F. Stout, Perfect dominating sets, *Congr. Numer.*, **78**, 1990, 187-203.

[43] Lothaire M., *Combinatorics on words*, Addison-Wesley, Reading, 1983.

[44] de Luca, A., On the combinatorics of finite words, *Theor. Comput. Sci.*, **218**, 1999, 13-39.

[45] Murty, K. G., C. Perin, A 1-Matching Blossom-Type Algorithm for Edge Covering Problems, *Networks*, **12**, 1982, 379-391.

[46] Murty, K. G., Network Programming, *Prentice Hall*, 1992.

[47] Punnen A. P., Y. P. Aneja, On k-sum optimization, *Operations Research Letters*, **18**, 1996, 233-236.

[48] Reinelt, G. The linear ordering problem: algorithms and applications. *Research and Exposition in Mathematics*, **8**, Heldermann Verlag, Berlin, 1985.

[49] Schiavinotto, T., Stützle, T., The linear ordering problem: instances, search space analysis and algorithms. *J. Math. Model. Algorithms*, **3**, 2004, 367–402.

[50] Woeginger, G., On minimizing the sum of k tardinesses, *Inform. Process. Letters*, **38**, 1991, 253-256