

Sümegi Mihály István

P-GRÁF ALAPÚ RENDSZER MEGBÍZHATÓSÁG SZÁMÍTÓ ALGORITMUS TERVEZÉSE

Témavezető: Dr. Orosz Ákos

Műszaki Informatikai Kar, Rendszer- és Számítástudományi
Tanszék

Intézményi Tudományos Diákköri Konferencia

2024

Pannon Egyetem



P-gráf alapú rendszer megbízhatóság számító algoritmus tervezése

*Készítette: Sümegi Mihály István
Villamosmérnök Bsc*

Műszaki Informatikai Kar, Rendszer- és Számítástudomá-nyi Tanszék

Témavezető: Dr. Orosz Ákos

A folyamat-hálózat szintézis egy fontos kutatás terület, az iparban sok területen alkalmazzák gyárak vagy szállító hálózatok modellezésére, de a leggyakoribb alkalmazási területe a vegyipar. Két gyakori megközelítése a problémának a matematikai programozás valamint a P-gráf keretrendszer, ami egy páros gráfként reprezentálja a folyamatot. Ez adja a dolgozat által használt módszer alapját is.

A P-gráf rendszerben a megbízhatóság számító algoritmusok többnyire átfogó kombinatorikai módszereken alapulnak, viszont az itt bemutatott algoritmus csak a közvetlen összeköttetések alapján dolgozik. A munka a korábban már prezentált direkt számítási algoritmust fejleszti tovább. Az algoritmus eddigi nagy limitációja az volt, hogy köröket tartalmazó gráfokra nem volt működőképes. Ennek javítására jelen dolgozat bemutat egy iteratív módszert, aminek segítségével megoldja a gráfban található visszacsatolások által okozott problémákat. Ez a megoldás több példán végrehajtott tesztek alapján a korábbi megoldásokkal megegyező megoldást ad, jellemzően hatékonyabb futási idővel.

Tartalomjegyzék

1. Bevezetés	4
2. Irodalmi áttekintés	5
2.1. P-gráf	5
2.2. Megbízhatóság	6
2.3. Hagyományos megbízhatóság számító algoritmus	6
3. Az algoritmus	7
3.1. Alapelv	7
3.2. Duplikációk a számítás során	8
3.3. Kritikus egységek	12
3.4. Körök	13
3.5. Implementáció	15
4. Példa	16
5. Összehasonlítás	21
6. Összefoglalás	24
Irodalomjegyzék	26

1. fejezet

Bevezetés

Folyamatok tervezése és szintézise során több indikátort is célszerű figyelembe venni. Jellemzően a legfontosabb szempont a költség vagy profit, azonban a hosszútávú tervezés során fontos tényező a rendszer megbízhatósága is. A megbízhatóság meghatározására több ismert megoldás is létezik már, melyek különböző elvek mentén közelítik meg a feladatot. Az azonban általánosan igaz, hogy nincs egységesen legjobb módszer, ami minden rendszer esetén hatékonyabb a többinél. Így mindig van lehetőség új alternatívák kidolgozására.

A munka folytatja egy korábban készített algoritmus, a közvetlen számításos módszer kidolgozását. Ez a módszer P-gráf segítségével ábrázolja a vizsgálandó folyamatokat, ezáltal képes kihasználni strukturális összefüggéseket. Már a P-gráf alkalmazásával is készültek megbízhatóság-meghatározó módszerek, azonban ezek mindegyike a működőképességi teszteken alapult, amelyek futtatása időigényes lehet. A közvetlen számításos módszer előnye, hogy megkerüli ezeket a tesztek, és a folyamat elejétől haladva, fokozatosan építi fel a rendszer megbízhatóságát.

Az algoritmus korábbi verziója ígéretes volt hatékonyság szempontjából, azonban hiányossága volt, hogy nem volt alkalmazható köröket tartalmazó folyamatokra. Jelen dolgozat az algoritmus kiegészítését mutatja be, amely már köröket tartalmazó rendszerekre is alkalmazható. A szükséges módosítások ismertetése után tesztpéldákon keresztül szemléltetésre kerül a módszer hatékonysága a korábbi algoritmusokhoz képest.

2. fejezet

Irodalmi áttekintés

2.1. P-gráf

A P-gráf keretrendszer [5] eredetileg vegyipari rendszerek modellezésre és szintézisére készült. A folyamathálózatokat páros gráfok segítségével modellezi, ahol a csúcsok két csoportja a műveleti egységek valamint az anyagok. A műveleti egységek az anyagok közötti átalakításokat jelölik.

Az anyagok további három kategóriára oszthatók. Az első kategória az alapanyagok, amelyek a folyamat bemenetét jelölik. Ezek mindig jelen vannak a hálózatban, nem szükséges a gyártásuk. A második típus a termékek, amelyek a folyamat eredményei. Ezeket tipikusan megadott mennyiségben és minőségben szükséges gyártani. Ezeken felül vannak még köztes anyagok, amelyek az átalakítások során előforduló köztes állapotokat jelölik.

Habár a P-gráf keretrendszert vegyipari folyamatok modellezésére készítették, ez a páros gráf alapú modellezési technika rengeteg feladatkör esetén alkalmazható. Az alkalmazások között fellelhető termelés ütemezés, ellátási lánc optimalizálás és fuvarszervezés is.

A P-gráf keretrendszer egy folyamatot, vagy egy szintézis feladatot formálisan a (P, R, O) hármassal ír le, ahol P a termékek halmaza, R a nyersanyagok halmaza, O pedig a műveleti egységek halmaza. A feladathoz tartozik az M halmaz, ami az összes lehetséges anyagot tartalmazza, ami a feladatban vagy rendszerben előfordul. Értelmszerűen $P \subset M$ és $R \subset M$. A műveleti egységek átalakításokat jelölnek, amelyek bizonyos bemeneti anyagokból bizonyos kimeneti anyagokat állítanak elő, megadott szabályok szerint. Ennek megfelelően a műveleti egységeket a bemeneti és kimeneti anyagaik segítségével határozzuk meg: $O \subseteq \mathcal{P}(M) \times \mathcal{P}(M)$. Egy $o \in O$ műveleti egységet jellemző-

en az (α_o, β_o) párral írunk le, ahol α_o jelöli a bemeneti anyagokat, β_o pedig a kimeneti anyagokat.

2.2. Megbízhatóság

Összetett folyamathálózatok megbízhatóságát gyakorlatban többnyire közelítési módszerekkel számolják, például Monte Carlo szimulációval [8] vagy Markov-láncok [10] segítségével. Zárt képletek csak egyszerűbb struktúrájú rendszerek esetén alkalmazhatóak [1].

A determinisztikus algoritmusok jelentős része vagy csak speciális struktúrájú rendszerek esetén alkalmazható [12], vagy kézi beavatkozást igényel [11]. További speciális esetnek minősül, ha csak lokális redundancia szerepel a rendszerben [9].

Általános hálózathoz determinisztikus algoritmusok továbbá a P-gráf rendszerben találhatóak módszerek [7] is, amelyek a modell felírása után kézi beavatkozás nélkül képesek működni. A dolgozat is P-gráf alapokon közelíti meg a megbízhatóság meghatározását.

2.3. Hagyományos megbízhatóság számító algoritmus

Az eddigi módszerek többsége ugyanazon az elven alapszik. Mivel a gráfban a minden műveleti egység két állapotban lehet: működő képes vagy nem, ez 2^n kombinációt hoz létre. Ezek a rendszer lehetséges állapotai. A műveleti egységek megbízhatóságai alapján (ezek megfelelő összesorzásával) minden állapot előfordulásának meghatározható a valószínűsége. A 2^n állapot egy teljes eseményteret alkot, így az összes valószínűség összege 1.

A korábbi, P-gráf alapú módszerek átfogó elve, hogy a rendszer állapotainak működőképességét vizsgálják. A módszerek alappillére a működőképességi teszt, amely egy állapotról meg tudja állapítani, hogy működőképes-e. Amennyiben ismerjük a működőképes állapotok halmazát, ezen állapotok valószínűségeinek összege kiadja a rendszer megbízhatóságát. Strukturális megbízhatóság esetén ez a teszt a P-gráf keretrendszer MSG algoritmusára vezethető vissza. Mivel a 2^n állapot egyenkénti vizsgálata sok idő lenne, így léteznek módszerek a számítások gyorsítására [4].

3. fejezet

Az algoritmus

3.1. Alapelv

Az új algoritmus a korábbiaktól eltérő megközelítést alkalmaz, amely nem támaszkodik a működőképességi tesztekre. Az algoritmus a rendszer alapvető valószínűség-számítási összefüggéseiből indul ki. Először tekintsük az egységeket a hálózatban független elemekként. Definiáljuk a P-gráf elemeinek kimeneti megbízhatóságát annak valószínűségeként, hogy az adott elem és annak minden előfeltétele adott a rendszer egy állapotában. Egy anyag esetében ez azzal egyezik meg, mintha a rendszer egyetlen terméke a kiválasztott anyag lenne, és így határoznánk meg a megbízhatóságot. Egy műveleti egység esetében az ekvivalens valószínűség akkor jön ki, ha a rendszer egyetlen termékét az adott műveleti egység gyártja, és semmi más. Jelöljük ezt a kimeneti megbízhatóságot a *csucs_neve* csúcs esetén P_{csucs_neve} formában.

Ebben a rendszerben egy anyag kimeneti megbízhatóságát felírhatjuk a következő módon: Legyen az anyagot gyártó műveleti egységek halmaza a $\phi^-(m)$, ahol az m az aktuális anyag. Mivel egy anyagnak a működési feltétele hogy legalább egy műveleti egység gyártsa, ebben az esetben a szokványos módszernek megfelelően kiszámoljuk annak az esélyét hogy egyetlen műveleti egység se gyártja az adott anyagot, majd ezt egyből kivonva a következő egyenletet kapjuk:

$$P_m = 1 - \prod_{o \in \phi^-(m)} (1 - P_o) \quad (3.1)$$

A műveleti egységeknél egy hasonló egyenlet felírható: Legyen az $\alpha(o)$ azon anyagok halmaza amelyek az adott o műveleti egységnek a bemenetei. Egy

műveleti egység működéséhez két dolog szükséges: (i) az összes általa felhasznált anyag jelen legyen a rendszerben, valamint (ii) az adott egység maga működőképes. Ezek alapján a kimeneti megbízhatósága az anyagok kimeneti megbízhatóságának szorzata szorozva a műveleti egység saját megbízhatóságával (amit r_o -val jelölünk).

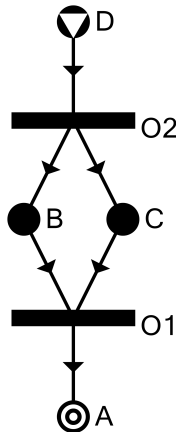
$$P_o = r_o \prod_{m \in \alpha(o)} P_m \quad (3.2)$$

Amíg egy csúcs bemenetei teljesen függetlenek egymástól, ezekkel az egyenletekkel egy P-gráfnak a megbízhatósága helyesen kiszámolható. Nyilvánvalóan ez egy nagyon limitált halmaz a lehetséges P-gráfoknak.

3.2. Duplikációk a számítás során

A duplikáció problémája

A előző részben bemutatott egyenletek azért nem működnek általános felépítésű gráfok esetén mert a számítás során előfordulhatnak ismétlődő tagok. Ezt a hibát a 3.1 példának a következő levezetése egyértelműen láthatóvá teszi.



3.1. ábra. Példa gráf a módosítás nélküli formulák helytelen működésére

Az előbbiek alapján a következő egyenletek írják le az ábrán látható gráfot.

$$\begin{aligned}
 P_A &= P_{O1} \\
 P_{O1} &= r_{O1}P_BP_C \\
 P_B &= P_{O2} \\
 P_C &= P_{O2} \\
 P_{O2} &= r_{O2}P_D \\
 P_D &= 1
 \end{aligned} \tag{3.3}$$

Ha ezekbe behelyettesítünk, akkor a következőt egyenletet kapjuk:

$$\begin{aligned}
 P_B &= P_C = P_{O2} = r_{O2} \\
 P_A &= P_{O1} = r_{O1}r_{O2}^2
 \end{aligned} \tag{3.4}$$

Az eredmény, amint látható $r_{O1}r_{O2}^2$ lett, ami egyértelműen hibás, hiszen a rendszer tényleges megbízhatósága $r_{O1}r_{O2}$. A hiba oka, hogy az $O2$ műveleti egységből az A termék több úton is elérhető. Ezt a jelenséget duplikációnak nevezzük.

Duplikáció keresés

Hogy ezt a problémát megoldjuk, első lépésben meg kell találnunk a duplikációkat. Ez egy gráf bejárásos algoritmus segítségével tehető meg. Az algoritmus a termékekből kiindulva a gráf minden eleméhez rendel egy számot, ami az adott csúcshoz a kimeneteihez rendelt számok összege. Egyenlőre az algoritmus egy csúcshoz csak akkor rendel értéket, ha minden kimenetéhez rendelt már értéket. Ez ciklikus gráfok esetén nem működik, erről a kiegészítésről a későbbiekben lesz szó.

Ennek az algoritmusnak a futásából kapunk egy gráfot ahol minden csúcshoz egy szám van rendelve. Ez a szám azt jelenti, hogy a csúcsból legalább annyi különböző úton érhető el termék, mint a hozzárendelt szám. Így minden olyan anyag, vagy műveleti egység, amihez a hozzárendelt szám egynél nagyobb, duplikációnak számít.

Megbízhatóság meghatározása

Miután meghatároztuk a duplikációkat a probléma megoldásához meg kell szüntetni a hatványokat a folyamat eredményében. Ennek eléréséhez a duplikációk megbízhatóságát nem szorozzuk be a részeredménybe hanem kiemelve tovább visszük a következő számításba. A gráf csúcsainak kimeneti megbízhatósága így nem egy egyszerű szám érték, hanem egy szimbolikus kifejezés, ami a kiemelt megbízhatóságok csoportjaihoz együtthatókat rendel. Jelöljük a kiemelt elemek halmazát S_n -el. Ekkor a kimeneti megbízhatóságot általánosságban a következőképpen írhatjuk fel:

$$\sum_{x \in \mathcal{P}(S_n)} a_x x \quad (3.5)$$

Mivel ez a formulát így felbontva kell kezelni, az egyes anyagokhoz és műveleti egységekhez nem tárolhatjuk ennek az eredményét pusztán egy számként. Arra van szükség, hogy az összes együtthatót külön tartsuk számon. A hatványhalmaznak akár rengeteg eleme is lehet, melyek nagy része nulla együtthatóval rendelkezik, és a sorba rendezésük sem triviális. Ez azt jelenti hogy az együtthatók tárolására egy hagyományos tömb nem a megfelelő adatszerkezet. A módszer egy hozzárendelés alapú adatszerkezetet használ, melyben az értelmezési tartomány hatványhalmaz elemei az értékészlete pedig a valós számok halmaza. Ez a hozzárendelés minden halmazhoz az együtthatóját rendeli hozzá. A gyakorlati megvalósításban ez egy asszociatív tárolóként van implementálva ahol a hatékonyság érdekében a nulla együtthatóval rendelkező halmazok nem kerülnek tárolásra.

$$P_n = \mathcal{P}(S_n) \mapsto \mathbb{R} \quad (3.6)$$

Ehhez a formátumhoz definiáljuk a műveleteket úgy, hogy a duplikációkat megelőzzük. Az összeadás és kivonás egyszerű, mivel szimplán elvégezzük a műveletet tagonként. Ha adottak az $A := \mathcal{P}(S_n) \mapsto \mathbb{R}$ és $B := \mathcal{P}(S_n) \mapsto \mathbb{R}$ hozzárendelések, akkor a $C = A \pm B$ összeg:

$$\begin{aligned} C &:= \mathcal{P}(S_n) \mapsto \mathbb{R} \\ C(z) &= A(z) \pm B(z), \quad \forall z \subseteq S_n \end{aligned} \quad (3.7)$$

A szorzás viszont meglehetősen komplexebb. Első lépéskét az egyik érték összes tagját megszorozzuk a másik érték minden tagjával. Ekkor a hagyományos szorzás során a kiemelt tagok hatványozásra kerülnének, viszont a cél pont ennek a megakadályozása. Így két elem összeszorozásakor csak az együtthatókat kell szorozni, és az eredményt a két kiemelt halmaz elemeinek az uniójához rendelni. Például kettő változó, x és y kiemelése esetén az alábbi ábra mutatja, hogy egyes tagok összeszorozása milyen típusú taghoz vezet.

	1	x	y	xy
1	1	x	y	xy
y	y	xy	y	xy
x	x	x	xy	xy
xy	xy	xy	xy	xy

3.2. ábra. A tagok alakulása szorzás során, kettő kiemelt változó esetén

A végső értéket az azonos csoportba eső eredmények összegzésével kapjuk meg.

Legyenek $A := \mathcal{P}(S_n) \mapsto \mathbb{R}$ és $B := \mathcal{P}(S_n) \mapsto \mathbb{R}$ hozzárendelések, melyek esetén a $C = A * B$ szorzat:

$$C := \mathcal{P}(S_n) \mapsto \mathbb{R}$$

$$C(z) = \sum_{\substack{s_1, s_2 \subseteq S_n \\ s_1 \cup s_2 = z}} A(s_1)B(s_2), \quad \forall z \subseteq S_n \quad (3.8)$$

A számításban továbbá szerepelnek konstans elemek. Ezekhez a műveletek egyszerűen definiálhatóak. Ha egy állandót hozzáadunk egy felbontott értékhez akkor az üres halmazhoz rendelt értékhez lesz hozzáadva. Szorzás esetén pedig minden együtthatóhoz hozzá kell szorozni. Alternatívaként egy konstans értéket tekinthetünk egy kiemelt értéként ahol a konstans értéke ez üres halmaz együtthatója lesz és minden más elem nulla.

Emellett az algoritmus helyes működéséhez a kiemelt műveleti egységek saját megbízhatóságát is kezelni kell. Az alap egyenletekben, mielőtt megkapjuk egy műveleti egység kimeneti megbízhatóságát, azt szorozni kell az egység saját megbízhatóságával. Ez nem duplikált műveleti egységeknél konstans érték, viszont duplikált egységeknek kiemelve kell kezelni, így a leképezésben is ennek

megfelelően kell kezelni. Mivel ez a értéket ki kell emelni a számításban, ezért a saját megbízhatóságát egy olyan hozzárendeléssel ábrázoljuk, ahol az üres halmaz együtthatója 0, helyette az aktuális egységhez tartozó együttható lesz fixen 1. Tehát az o duplikált műveleti egység eredeti megbízhatósága r_o , viszont az új módszerben használt képletekben az R_o jelölést használjuk innentől ($R_o := \mathcal{P}(S_n) \mapsto \mathbb{R}$), ahol

$$R_o(H) = \begin{cases} 1, & \text{ha } H = \{o\} \\ 0, & \text{egyébként} \end{cases} \quad (3.9)$$

Ha a megbízhatósági számítások során a csúcsokhoz nem csak egy-egy számot határozunk meg, hanem egy ilyen $\mathcal{P}(S_n) \mapsto \mathbb{R}$ függvényt, amelyekkel a fent definiált módon végezhetünk műveleteket, akkor végső megbízhatósági értékünk szintén egy ilyen függvény lesz, jelölje ezt $\mathcal{R}$. A rendszer végső megbízhatósága ebből a függvényből kiszámolható, hiszen itt már a kapott értékek beszorozhatóak a kiemelt műveleti egységek megbízhatóságaival. A rendszer R megbízhatósága így:

$$R = \sum_{s \subseteq S_n} \left(\mathcal{R}(s) * \prod_{o \in s} r_o \right) \quad (3.10)$$

3.3. Kritikus egységek

A kritikus egységek olyan műveleti egységek amik mindenképpen szükségesek a hálózat működéséhez. Ha bármely kritikus egység nem működik akkor nem lehet működőképes a rendszer. Ezt a tulajdonság felhasználható a duplikációk csökkentésére. Ha egy kritikus egység duplikáció, akkor a közvetlen számításban kiemelten kezelve úgy jelenik meg hogy a végeredményben, hogy minden tagban szerepel. Ezt a jelenséget ki tudjuk használni a duplikációk csökkentésére. Ha a számítás során a kritikus egységek megbízhatóságát egynek tekintjük akkor nem szükséges duplikációként kezelni, mivel az egynek az összes hatványa egy, így a számításban rejtve keletkező hatványozások ilyen esetben nem okoznak hibát. Kihasználva azt, hogy a kritikus műveleti egység minden tagban szerepel, ahhoz, hogy megkapjuk a helyes eredményt a rendszer megbízhatóságára, a számítás végén a végeredményt még meg kell szorozni az

összes kritikus egység megbízhatóságával.

A kritikus egységek keresése jelenleg az MSG algoritmus segítségével történik. Ennek a folyamata egyszerű: a gráfot leteszteljük minden esetre ahol csak egyetlen műveleti egység van meghibásodva, ha egy ilyen esetben a gráf nem működőképes akkor az abban az esetben a kiválasztott egység egy kritikus egység.

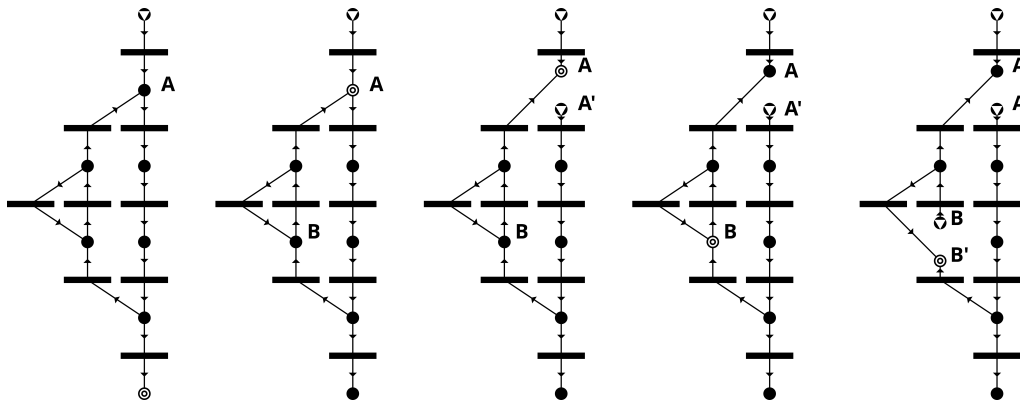
3.4. Körök

A módszer korábbi verziójának korlátja, hogy köröket tartalmazó P-gráfok esetén nem működőképes, hiszen mind a duplikáció keresés, mint a számítás épít arra, hogy egy csúcshoz eljutva annak már minden előzményét ki tudta számolni. Jelen dolgozat újdonsága a módszer kiegészítése, hogy kört tartalmazó hálózatokra is alkalmazható legyen. Először is a duplikáció keresési algoritmust ki kell egészíteni, hogy működjön köröket tartalmazó gráfokon is. Ez egyszerű probléma mivel az algoritmus jelenleg csak akkor számolja ki az értéket egy csúcshoz, ha annak minden kimenetéhez már tartozik érték. Ez a folyamat elakad, ha a hálózat tartalmaz kört, mivel ebben az esetben lesznek csúcsok, amelyek körkörösén függenek egymástól, és egyikhez sem határozható meg az érték. Mivel egy körnek minden tagja duplikáció, így ezek a csúcsok, amelyekben a duplikáció keresés elakad, automatikusan duplikációnak vehetőek. Továbbá, ha van olyan csúcs a gráfban, amely nem része körnek, de elérhető belőle kör, akkor az is duplikációnak tekintendő. Így a duplikáció kereső algoritmust módosítani nem kell. Amennyiben a számítás megakad, mert minden hátralévő csúcsnak van még nem kiszámolt kimenete, akkor minden kimaradt csúcs duplikációnak tekinthető.

A számításban a köröket viszont jelentősen nehezebb megoldani. A módosított módszer körhöz érve két lépést hajt végre. A teljes folyamatot a 3.3 ábra illusztrálja. Az első lépés a kör felbontása. Ezt jellemzően anyag csúcsban egyszerűbb megtenni, de a leírt elvekhez hasonlóan műveleti egységre is működik. Ahogy korábban volt róla szó, egy anyag kimeneti megbízhatósága azzal ekvivalens, mintha a rendszer megbízhatóságát határoznánk meg úgy, hogy ez az anyag az egyetlen termék. Ha a gráfot módosítjuk erre a formára, utána az anyag csúcsnál elvégezzük a felbontást (A csúcs a 3.3. ábra második lépésén).

Minden műveleti egység, ami ezt az anyagot gyártotta, ezt továbbra is megteszi. Azonban azok a műveleti egységek, amelyek az anyagot felhasználták, helyette egy új nyersanyagot használnak.

Mivel a rendszerben továbbra is az A csúcs az egyetlen termék, így a teljes leszámhlálásos megbízhatóság számítását használva igazolható, hogy a felbontás előtti és felbontás utáni rendszer megbízhatósága megegyezik (a működőképes állapotok halmaza ugyanaz). Ha a felbontott gráfban meghatározzuk a megbízhatóságot, az az eddigiekből következően az eredeti rendszerben meg fog egyezni az A csúcs kimeneti megbízhatóságával. Miután az A csúcs kimeneti megbízhatóságára már ismert a helyes érték, akkor a közvetlen számításos módszer ebből kiindulva a ráépülő csúcsokhoz is helyesen meg tudja ezt határozni.



3.3. ábra. P-gráf felbontásának folyamata

Amennyiben a felbontott gráfban is található még kör, és az algoritmus ismét elakad, akkor ezt a kört is fel kell bontani ugyanezen szabályok mentén (a példában a B csúcsnál). A belső felbontás után a B csúcs kimeneti megbízhatósága alapján az A csúcshoz is meghatározható az érték, amit aztán fel lehet használni az eredeti gráfban. Fontos azonban megjegyezni, hogy a B csúcsához számított érték csak lokálisan, a szétbontott hálózatban volt helytálló, de az eredeti rendszert tekintve nem feltétlenül, így azt újra ki kell majd számolni akkor is, mikor az eredeti rendszerben odaér a számítás.

3.5. Implementáció

Az Implementáció kissé eltér az elmélettől mivel minden körhöz szintetizálni egy új gráfot lassú lenne. A körök megoldásának logikája előtt viszont ismerni kell az alap logikáját a megvalósításnak. Az algoritmus számon tartja a kiszámolásra kész egységek halmazát valamint a már kiszámolt egységek halmazát. A kiszámolt egységek halmaza kezdetben az alapanyagok halmaza, a kiszámolható egységek halmaza pedig azokkal az egységekkel amik, csak alapanyagokat használnak fel. Az algoritmus először kivessz egy elemet a kiszámolásra kész listából majd kiszámolja azt, és hozzáadja a kiszámolt halmazhoz. Ezután ellenőrzi, hogy a csúcs kimenetei között van-e olyan, aminek így már minden bemenetéhez tartozik kiszámított érték. Amennyiben igen, akkor hozzáadja a kiszámolhatók listájához. Ezt a folyamatot addig ismétli amíg minden egység be nem kerül a kiszámolt halmazba.

Ezt a logikát bővítjük ki a felbontott gráfokhoz. Viszont ezek a gráfok nem jönnek ténylegesen létre az algoritmus futása során, hanem a teljes számítás egy gráfon történik. Amikor az algoritmus amikor nem tud újabb csúcsot biztosan meghatározni, akkor választ egy meg nem határozott anyagot, egynek veszi a megbízhatóságát (ezzel imitálva a gráf felbontását), és beveszi a kiszámoltak listájába. A számítást ezek után a szokásoshoz hasonlóan folytatja, de kicsit módosítva. Az algoritmus számon tartja, hogy mely csúcsok értéke függ egy hasonló feltételezéstől, hiszen az ezekhez kiszámolt érték még nem feltétlenül helyes. Amikor egy ilyen "megjelölt" csúcshoz később visszaér az algoritmus, és a kimeneti megbízhatósága új értékre frissül, akkor a kimeneteit is újra kell számolni. A megállási feltétele az algoritmusnak, hogy semelyik csúcs újbóli kiszámítása sem okozott már változást.

Ez a módszer jó eredményt ad, de hatékonyságát tekintve nem tökéletes, mivel előfordulnak olyan esetek amikor fölösleges kétszer számol egy részt mert túl korán elindult egy úton, ez majd a példában látható lesz. De végeredményben ez a folyamat ugyanaz számítást adja, mint ami a gráf tényleges felbontása esetén is kijönne.

4. fejezet

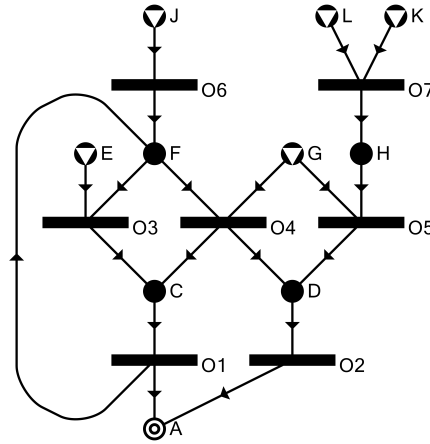
Példa

Mivel a példában nem egy konkrét valós rendszert elemzünk ezért az egyszerűség kedvéért minden műveleti egységnek 99% lesz a megbízhatósága, tehát a valószínűség értéke 0.99 lesz minden esetben.

A példák során a kimeneti megbízhatóságok függvényeit az egyszerűbb formalizálás érdekében speciális tartalmú vektorokként fogjuk ábrázolni, helytakarékosság miatt ritkás formában, tehát azon halmazok, melyeknek az együtthatója nulla, nem kerülnek felírásra. Például, ha $S_n = \{O1, O2\}$, vagyis $\mathcal{P}(S_n) = \mathcal{P}(\{O1, O2\}) = \{\emptyset, \{O1\}, \{O2\}, \{O1, O2\}\}$, és az egyes részhalmazokhoz rendelt értékek rendre 0; 0,54; 0; és 0,96, akkor ezt a következő vektor fogja ábrázolni:

$$\left[\begin{array}{cc} \{O1\} & \mapsto 0,54 \\ \{O1, O2\} & \mapsto 0,96 \end{array} \right]$$

A példa során a 4.1 ábrán látható P-gráf lesz megoldva. Ehhez az első lépés a duplikációk meghatározása. A duplikáció keresés kiindul a A terméktől, majd a $O2, D, O5, H, O7$ csúcsokat végig tudja járni. Az $O1, O4$, illetve G csúcsok egyikébe sem képes tovább haladni, mivel vannak olyan kimeneteik, amelyekhez még nem rendelt értéket. Így az algoritmus itt leáll, és duplikáció lesz minden olyan műveleti egység, amelynek az értéke legalább 2, vagy amelyhez az algoritmus nem jutott el. A példára tehát az $O1, O3, O4$, és $O6$ egységek duplikációk. Az $O2, O5$ és $O7$ viszont nem, mivel csak egy úton lehet elérni őket.



4.1. ábra. A példa ábrázolása P-gráf segítségével

Miután megvannak a duplikációk, elkezdődik a megbízhatóság kiszámolása. Első lépésként elkezdjük azoknak a műveleti egységeknek és anyagoknak a kimeneti megbízhatóságát kiszámolni amelyeknek minden előzményét ismerjük. A nyersanyagok kimeneti megbízhatóságai mindig 1-ek:

$$P_E = P_G = P_J = P_L = P_K = 1 \quad (4.1)$$

Az első számítás az $O6$ műveleti egység, melynek eredménye triviális:

$$P_{O6} = P_J R_{O6} = \left[\{O6\} \mapsto 1 \right] \quad (4.2)$$

Ezt követően az $O7$, H és $O5$ Lánc kerül kiszámolásra ezek szintén egyszerűek, mivel egyik elem sem duplikáció.

$$\begin{aligned} P_{O7} &= P_L P_K R_{O7} = \left[\emptyset \mapsto 0.99 \right] \\ P_H &= P_{O7} \\ P_{O5} &= P_H P_G R_{O5} = \left[\emptyset \mapsto 0.9801 \right] \end{aligned} \quad (4.3)$$

Itt az algoritmus megakad, mivel nincsen több tag aminek minden előfeltétele meg lenne. Ilyenkor egy elem, aminek nem minden előzménye ismert feltételezésre kerül. Legyen ez most a F anyag, amihez feltesszük, hogy az $O1$ kimeneti megbízhatósága 1. Ilyenkor a számolást ugyanúgy kell elvégezni, mint a normál esetben, annyi különbséggel, hogy tudjuk, az $O1$ értéke nem helyes.

Ezt követően ez $O3$ és $O4$ egységeket a szokásos módon kilehet számolni.

$$\begin{aligned}
 P_{O1} &= \left[\begin{array}{l} \{O1\} \mapsto 1 \end{array} \right] \\
 P_F = 1 - (1 - P_{O1})(1 - P_{O6}) &= \left[\begin{array}{l} \{O1\} \mapsto 1 \\ \{O6\} \mapsto 1 \\ \{O1, O6\} \mapsto -1 \end{array} \right] \\
 P_{O3} = P_E P_F R_{O3} &= \left[\begin{array}{l} \{O1, O3\} \mapsto 1 \\ \{O3, O6\} \mapsto 1 \\ \{O1, O3, O6\} \mapsto -1 \end{array} \right] \\
 P_{O4} = P_G P_F R_{O4} &= \left[\begin{array}{l} \{O1, O4\} \mapsto 1 \\ \{O4, O6\} \mapsto 1 \\ \{O1, O4, O6\} \mapsto -1 \end{array} \right]
 \end{aligned} \tag{4.4}$$

A következő lépésben a jelenlegi megvalósításban kiszámolja a D és $O2$ csúcsokat, ami a gráfot látva felesleges művelet, hiszen nem részei a körnek, azonban ez nem okoz hibás működést, csupán egy kicsivel lassabb működést. Ez javítható lenne a köröknek a tényleges detektálásával, elkerülve a felesleges műveleteket, ez azonban majd a jövőbeli fejlesztések során fog megvalósulni. A D és $O2$ után az algoritmus megy tovább a C és a $O1$ elemekre is.

$$\begin{aligned}
 P_D = 1 - (1 - P_{O4})(1 - P_{O5}) &= \begin{bmatrix} \emptyset \mapsto 0.9801 \\ \{O1, O4\} \mapsto 0.0199 \\ \{O4, O6\} \mapsto 0.0199 \\ \{O1, O4, O6\} \mapsto -0.0199 \end{bmatrix} \\
 P_{O2} = P_D R_{O2} &= \begin{bmatrix} \emptyset \mapsto 0.970299 \\ \{O1, O4\} \mapsto 0.019701 \\ \{O4, O6\} \mapsto 0.019701 \\ \{O1, O4, O6\} \mapsto -0.019701 \end{bmatrix} \\
 P_C = 1 - (1 - P_{O3})(1 - P_{O4}) &= \begin{bmatrix} \{O1, O3, O4\} \mapsto -1 \\ \{O1, O3, O4, O6\} \mapsto 1 \\ \{O3, O4, O6\} \mapsto -1 \\ \{O1, O4\} \mapsto 1 \\ \{O4, O6\} \mapsto 1 \\ \{O1, O4, O6\} \mapsto -1 \\ \{O1, O3, O6\} \mapsto -1 \\ \{O3, O6\} \mapsto 1 \\ \{O1, O3\} \mapsto 1 \end{bmatrix} \\
 P_{O1} = P_C R_{O1} &= \begin{bmatrix} \{O1, O3, O4\} \mapsto -1 \\ \{O1, O4\} \mapsto 1 \\ \{O1, O3\} \mapsto 1 \end{bmatrix}
 \end{aligned} \tag{4.5}$$

Ezen a ponton az algoritmus körbe ért, de a számítás során az $O1$ értéke megváltozott (korábban a feltételezés miatt 1 volt), ezért az F csúcs kimeneti megbízhatóságát is újra kell számolni. Ez addig folytatódik, amíg az értékek nem stabilizálódnak.

$$P_C = 1 - (1 - P_{O3})(1 - P_{O4}) = \left[\begin{array}{ll} \{O1, O3, O6\} & \mapsto -1 \\ \{O3, O4, O6\} & \mapsto -1 \\ \{O1, O3, O4, O6\} & \mapsto 1 \\ \{O6\} & \mapsto 1 \\ \{O1, O3, O4\} & \mapsto -1 \\ \{O1, O4\} & \mapsto 1 \\ \{O1, O3\} & \mapsto 1 \end{array} \right] \quad (4.6)$$

Ezt követően az eredmények az $O3$ és $O4$ -re azonosak, ezért ebben az irányban a számítás befejeződik. Így mivel az A terméknek már minden termelőjének ki van számolva a kimeneti megbízhatósága, az A maga is kiszámolható, és a következő eredményt adja.

$$P_A = 1 - (1 - P_{O1})(1 - P_{O2}) = \left[\begin{array}{ll} \emptyset & \mapsto 0.970299 \\ \{O4, O6\} & \mapsto 0.019701 \\ \{O1, O4, O6\} & \mapsto -0.019701 \\ \{O1, O3, O4\} & \mapsto -0.029701 \\ \{O1, O4\} & \mapsto 0.029701 \\ \{O1, O3\} & \mapsto 0.029701 \end{array} \right] \quad (4.7)$$

Az A csúcs kimeneti megbízhatóságát a (3.10) egyenletbe behelyettesítve a kapott megbízhatósági értékre 0.999893139102 jön ki, ami megegyezik a korábbi algoritmusok eredményeivel.

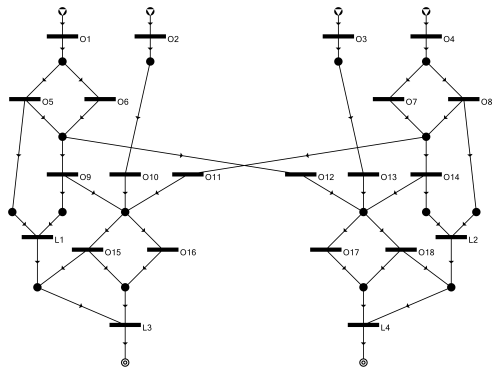
5. fejezet

Összehasonlítás

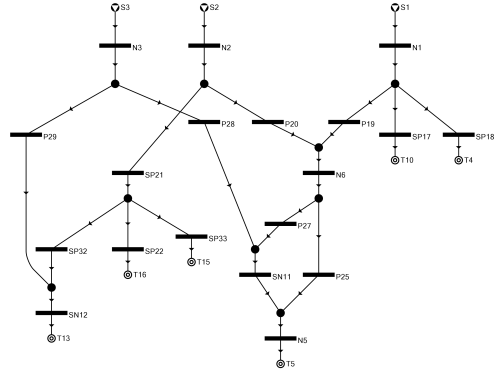
A új algoritmust teljesítményének tesztelésére kilenc különböző rendszert ismertetünk. Hasonlítási pontként a korábbi SSGR valamint BDD [4] algoritmusok lesznek használva. A részletes összehasonlítás érdekében a hálózatok különböző méretűen és komplexitásúak, valamint vannak köztük kört tartalmazó, és nem tartalmazó rendszerek is.

Az egyik példa a szemléltetésre is használt (4.1. ábrán látható) hálózat. A többi hálózatot a 5.1. és 5.2. ábrák mutatják. A 5.1(a) ábra egy repülőgép üzemanyag továbbító hálózatát ábrázolja [3, Chapter 14]. A 5.1(b) ábrán látható hálózat egy gázhálózat modellje [7]. A 5.1(c) ábra a vegyiparban ismerős HDA folyamat [2] modellje, kiegészítve redundáns lehetőségekkel. A példák között szerepel továbbá egy komplex folyamat adipinsav előállítására 5.1(d), valamint két összetett hálózat (5.1(e) és 5.1(f)), amelyek a nemrég megjelent könyvben találhatóak [6]. Ezen kívül adott még egy szintetikus példa a 5.2(a). ábrán, valamint a 5.2(b) ábrán egy komplex vegyipari rendszer modellje [5].

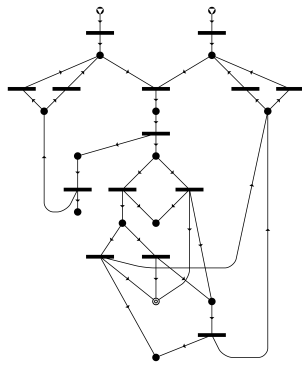
A tesztek egy i3-1005G1 processzoron, 8GB memóriával lett tesztelve. A Folpet példa (5.2(a) ábra) kivételével minden teszt százszor lett futtatva. Mivel a folpet példának több perc a futási ideje ezért csak kétszer lett futtatva a teszt. A táblázatban látható számok ezeknek az eredményeknek az átlagai.



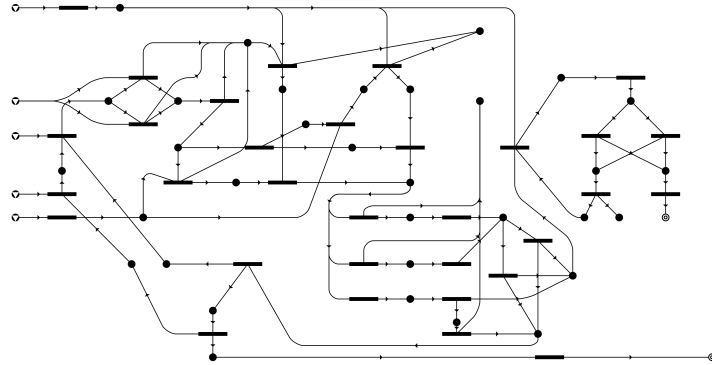
(a) Repülőgép hajtómű modell



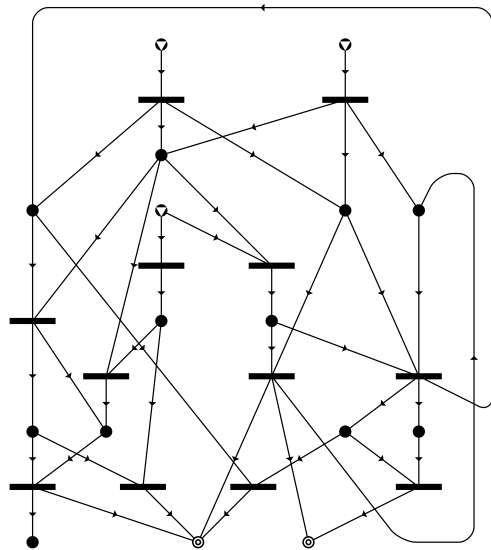
(b) Gázhálózat model



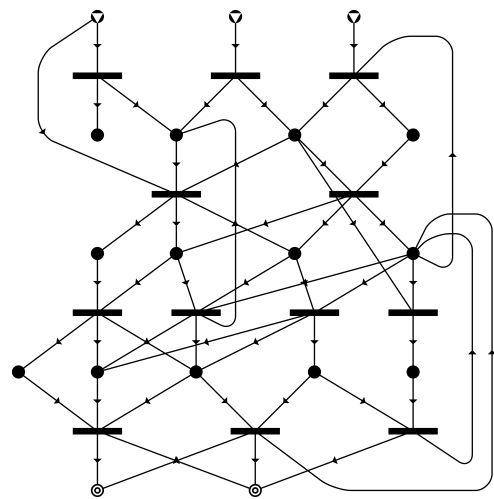
(c) HDA folyamat



(d) Adipic acid structure

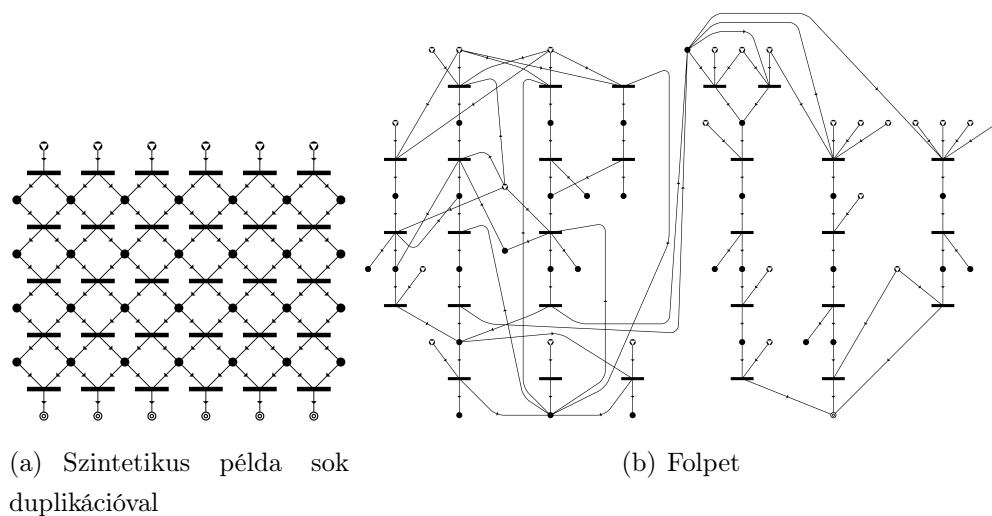


(e) Összetett példa



(f) Összetett példa

5.1. ábra. Példák



5.2. ábra. Példák folytatás

Gráf	BDD	SSGR	Direkt Számítás
Minta példa 4.1 ábra	2067 ms	2061.35 ms	1.557 ms
Repülőgép hajtómű 5.1(a) ábra	6.858 ms	5.125 ms	0.2245 ms
Gázhálózat 5.1(b) ábra	0.1256 ms	0.148 ms	0.07287 ms
HDA 5.1(c) ábrák	6.043 ms	16.46 ms	47.94 ms
Adipic acid structure 5.1(d)	7793.7 ms	4619.6 ms	152.21 ms
Könyv 7.1 5.1(e) ábra	7.544 ms	14.31 ms	0.2369 ms
Könyv 8.1 5.1(f) ábra	10.39 ms	17.32 ms	8.623 ms
Folpet 5.2(a) ábra	578983.2 ms	15499 ms	54.493 ms
Szintetikus példa 5.2(b) ábra	666.09 ms	153.61 ms	123.23 ms

5.1. táblázat. Az új algoritmus összehasonlítása a régiekkel 9 teszt példa esetén

Mint látható az algoritmus jelentős felgyorsulást ért el a HDA 5.1(d) kivételével ez valószínűleg a több körnek valamint az ezekből következő sok duplikációnak köszönhető. Az eredmények között különösen meglepő a szintetikus példa eredménye 5.2(b), ahol a sok duplikációnak köszönhetően az elvárt eredmény a lassulás volt, azonban az a tesztek nem ezt mutatják. Az részben elképzelhető, hogy a kritikus műveleti egységek kiszűrésének köszönhető, de a pontos oka további vizsgálatokat igényel.

6. fejezet

Összefoglalás

A dolgozat ismertette a korábbi munka során készített közvetlen számításos algoritmus kiegészítését köröket tartalmazó hálózatokra. Az algoritmus által adott eredmény a tesztek alapján helyes, és hatékonyság tekintetében továbbra is a legtöbb rendszer esetén az élen jár a korábbi módszerekkel összehasonlítva.

A módszer jelenlegi formájában exponenciális bonyolultságú a duplikált egységek számára tekintve. Azonban a leképezések ritkás kezelésének köszönhetően a rendszer komplexitása lokalizálódik az egyes számításokra, ezáltal még összetett rendszerek esetén is képes hatékonyabb működésre, mint a korábbi algoritmusok. Mindazonáltal még sok lehetőség van az algoritmus optimalizálására, ilyen például a duplikációk kiemelésének a további lokalizálása, valamint a körök esetén a hatékony bejárési sorrend megkeresése.

Irodalomjegyzék

- [1] Alessandro Birolini. *Reliability engineering*. Springer, 5 edition, 2007.
- [2] James Merrill Douglas. Conceptual design of chemical processes. 1988.
- [3] Federal Aviation Administration. *Aircraft Fuel System*, chapter 14. US, 2012.
- [4] M. Ferenczi. Megbízhatóság hatékony számítása folyamathálózatokban. In *ITDK 2022, Pannon Egyetem*, 2022.
- [5] F. Friedler, K. Tarjan, Y.W. Huang, and L.T. Fan. Combinatorial algorithms for process synthesis. *Computers & Chemical Engineering*, 16:S313–S320, 1992. European Symposium on Computer Aided Process Engineering—1.
- [6] Ferenc Friedler, Ákos Orosz, and J. Pimentel Losada. *P-graphs for process systems engineering*. Springer, 2022.
- [7] Z. Kovacs, A. Orosz, and F. Friedler. Synthesis algorithms for the reliability analysis of processing systems. *Central European Journal of Operations Research*, 27(2):573–595, Jun 2019.
- [8] Hiromitsu Kumamoto, Kazuo Tanaka, and Koichi Inoue. Efficient evaluation of system reliability by monte carlo method. *IEEE Transactions on Reliability*, R-26:311–315, 1977.
- [9] Hiroaki Mohri and Jun-ichi Takeshita. Graph reliability evaluation via random k-out-of-n systems. *Communications in Statistics-Theory and Methods*, pages 1–11, 2022.

- [10] S.J Prowell and J.H Poore. Computing system reliability using markov chain usage models. *Journal of Systems and Software*, 73(2):219–225, 2004. Applications of statistics in software engineering.
- [11] Enno Ruijters and Mariëlle Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15-16:29–62, 2015.
- [12] Ming Jian Zuo, Zhigang Tian, and Hongzhong Huang. An efficient method for reliability evaluation of multistate networks given all minimal path vectors. *IIE Transactions*, 39:811 – 817, 2007.