

TDK-dolgozat

Ferenczi Martin

Ferenczi Martin

4. évfolyam

MEGBÍZHATÓSÁG HATÉKONY SZÁMÍTÁSA FOLYAMATHÁLÓZATOKBAN

Témavezető: Orosz Ákos, tanársegéd

Pannon Egyetem

Műszaki Informatikai Kar

Rendszer- és Számítástudományi Tanszék

Tartalom

1	Bevezetés.....	5
2	P-gráf alapú megbízhatóság számítás.....	6
2.1	Az SSG algoritmus	8
2.2	Teljes bejárásos algoritmus	9
2.3	Bináris Döntési Diagram	11
3	Tabulistas algoritmus.....	15
4	SSGR (Solution Structure Generator for Reliability).....	18
4.1	Az SSGR algoritmus	20
4.2	Az algoritmus személtetése	21
5	Teszt eredmények.....	24
5.1	Szemléltető példa.....	24
5.2	Repülőgép üzemanyag rendszer	25
6	Összegzés	28
7	Irodalomjegyzék.....	29

Kivonat

Folyamathálózatok értékelése, rangsorolása során több fontos indikátor is van. A tervezés során az elsődleges szempont általában a hálózat költsége, vagy a megtermelt profit. Más gyakran használt jellemzők közé tartozik a fenntarthatóság, a megbízhatóság vagy a flexibilitás. Bizonyos indikátorok, mint a költség vagy a fenntarthatóság egyértelműen és gyorsan meghatározhatóak az előre definiált képletek segítségével. Vannak azonban olyan jellemzők, mint a megbízhatóság, amihez kombinatorikus jellege miatt nem adható meg egyszerű, gyors formula. Így a megbízhatóság pontos meghatározása általános esetben nagy számítási kapacitást igényel. A folyamathálózat szintézis témaköréhez kapcsolódóan még hiányzik az a módszer, amely egy tetszőleges rendszer megbízhatóságát precízen és gyorsan meghatározza, függetlenül annak komplexitásától.

Jelen munka egy olyan meglévő, a P-gráf keretrendszeren alapuló algoritmusból indul ki, amely képes folyamathálózatok megbízhatóságának precíz meghatározására. Bár az említett algoritmus általánosan használható, a jellege miatt magas a számítási igénye. A dolgozat célja tehát az algoritmus továbbfejlesztése a hatékonyság növelése érdekében.

A dolgozat több módosítás megvalósítását és tesztelését mutatja be. A módosítások 3 csoportba sorolhatóak. Az első csoport az eredeti algoritmus hatékonyságának közvetlen továbbfejlesztéseit tartalmazza. A másik csoportba az algoritmus Bináris Döntési Diagram elvű átírása, és annak javítása tartozik. A harmadik csoport a P-gráf keretrendszer SSG algoritmusán alapuló számítási módszer különböző verzióiból áll. Az ismertetett algoritmusok implementálása után a hatékonyság összehasonlítása több teszt példán keresztül történt, amelyek eltérnek mind jellegben, mind méretben. Az elvégzett tesztek igazolják az újabb algoritmusok hatékonyságát az eredetivel szemben, valamint megmutatják az új, javasolt algoritmusok egymáshoz képesti kapcsolatát.

1 Bevezetés

A napjainkban folyó kutatások között nagy szerepet kap a műszaki rendszerek tervezésével, elemzésével, optimalizálásával foglalkozó témakör. Egy műszaki rendszer vagy folyamathálózat tervezése során több szempontot is célszerű figyelembe venni. A leggyakrabban használt mérőszám a költség, hiszen ez az egyik fő motiváló erő a rendszer tervezése során, azonban a gyakorlatban komoly jelentősége van egyéb mutatóknak is, mint a fenntarthatóság, az irányíthatóság, illetve a megbízhatóság. Az utóbbi évtizedek alatt jelentős mennyiségű publikáció született mindegyik fontosabb mutató kapcsán, viszont ez korántsem jelenti azt, hogy ezen a területen már mindent megoldottak. Épp ellenkezőleg, minél több kutatás folyik a témában, annál több új igény, feladat, kihívás bukkan fel.

Jelen munka során a megbízhatósági indikátor került előtérbe. A megbízhatóság főbb nehézsége, hogy általános esetben nem létezik rá egyszerűen használható zárt képlet, így ennek az integrálása a rendszerek tervezése során komoly kihívást jelenthet. Ennek köszönhetően jellemzően a rendszerek tervezése és a megbízhatósági analízis két külön lépésre bomlik, azonban egy adott rendszer megbízhatóságának meghatározása is nehéz feladat.

A megbízhatóság meghatározására az iparban leggyakrabban szimulációs, vagy közelítő módszereket alkalmaznak, amelyek nem pontos értéket határoznak meg. Ezek közül a legismertebb szimulációs módszer a Monte Carlo szimuláció [1], de gyakori a Markov-láncok segítségével történő modellezés is [2]. A megbízhatóság meghatározását jelentősen egyszerűsíti, ha a rendszer felépítése valami egyszerűbben kezelhető struktúrán alapul. Például soros-párhuzamos rendszerekre és egyéb ismert struktúrákra léteznek ismert képletek [3]. Általános felépítésű rendszerre determinisztikus algoritmus a P-gráf keretrendszerre építve adható [4]. Ezeken felül gyakran alkalmaznak közelítő vagy sztochasztikus módszereket [5].

Jelen munka a P-gráf alapú megbízhatóság-meghatározó módszert veszi alapul, és a cél a módszer számítási hatékonyságának növelése. Ennek megfelelően a dolgozat először ismerteti a P-gráf keretrendszert és az arra építő megbízhatóság-számító algoritmust, majd ismerteti a kettő gyorsítási javaslatot. Végezetül az elvégzett tesztek összehasonlítják az új algoritmusokat a meglévőkkel, igazolva a gyorsítás mértékét.

2 P-gráf alapú megbízhatóság számítás

A P-gráf keretrendszer [6] páros gráfok segítségével ábrázolja a folyamat-hálózatokat, ahol a csúcsok két csoportját a műveleti egységek, valamint az anyagok alkotják. A csúcs típus megnevezése a vegyipari termelő rendszerekből jön, amelyek tervezésére a P-gráf módszertant eredetileg megalkották. Az anyagok a rendszerben használt, illetve előállított anyagokat jelölik. Ezek között szerepelnek a termékek, amiket a termelő üzem előállít és elad, a nyersanyagok, amelyek beszerzésével kezdődik a termelő folyamat, valamint a folyamat során létrehozott köztes anyagok és melléktermékek. A műveleti egységek az anyagok közötti átalakítást végző berendezések. A terminológia ellenére a P-gráf keretrendszer nem csak vegyipari rendszerek modellezésére használható. A bevezetése óta eltelt több, mint 30 év alatt több egyéb területen is alkalmazásra került, mint a víztisztító hálózatok [7], ellátási hálózatok [8], jármű-hozzárendelési feladatok [9], evakuációs útvonal-tervezés [10], vagy a teszt-szekvencia optimalizálás [11].

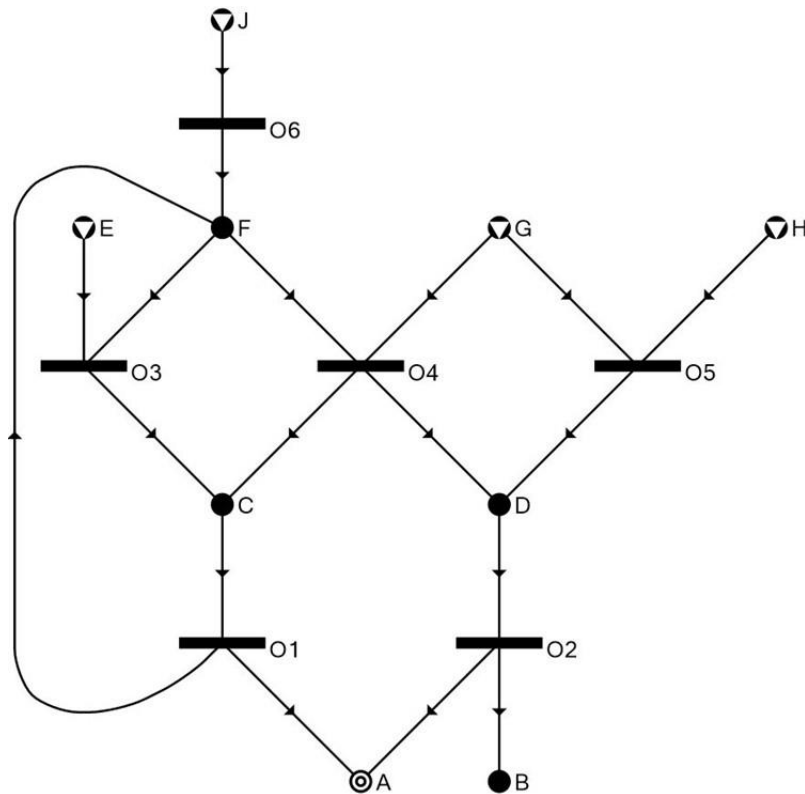
A P-gráf keretrendszerben a folyamatok, rendszerek, illetve szintézis folyamatok modellezése során az anyagok halmazát M jelöli, aminek részei a nyersanyagok ($R \subseteq M$), és a termékek ($P \subseteq M, R \cap P = \emptyset$). Ugyanitt a műveleti egységek halmazát O jelöli, amelyben minden műveleti egységek az általuk felhasznált és előállított anyagok segítségével definiálhatóak: $O \subseteq \wp(M) \times \wp(M)$. Egy műveleti egységet, legyen ez $o \in O$, egy pár határoz meg, $o = (\alpha, \beta)$, amelynek első eleme (α) a műveleti egység által felhasznált anyagok halmaza, míg második eleme (β) a műveleti egység által előállított anyagok halmaza.

Folyamat-hálózat szintézis feladatok esetén fontos a strukturálisan lehetséges hálózat fogalma. Strukturálisan lehetséges hálózatnak azokat a hálózatokat nevezzük, amelyek kizárólag a hálózat struktúráját (a tartalmazott műveleti egységeket és kapcsolataikat) tekintve, (i) képes minden terméket előállítani, és (ii) nem tartalmaz felesleges, a hálózat többi része számára használhatatlan elemeket. A P-gráf keretrendszer a strukturálisan lehetséges hálózatokat, más néven megoldás-struktúrákat, 5 axióma segítségével definiálja; egy hálózat akkor és csak akkor megoldás struktúra, ha teljesíti mind az 5 axiómát:

- Minden termék szerepel a hálózatban,
- Egy anyag csúcsot akkor és csak akkor nem gyárt műveleti egység, ha az anyag csúcs nyersanyagot jelöl,
- Csak a definiált műveleti egységek szerepelnek a hálózatban,
- A hálózat minden műveleti egységétől megy legalább 1 irányított út valamelyik termékbe,
- A hálózatban csak olyan anyag csúcsok vannak, amelyek valamely műveleti egységnek bemenetei vagy kimenetei.

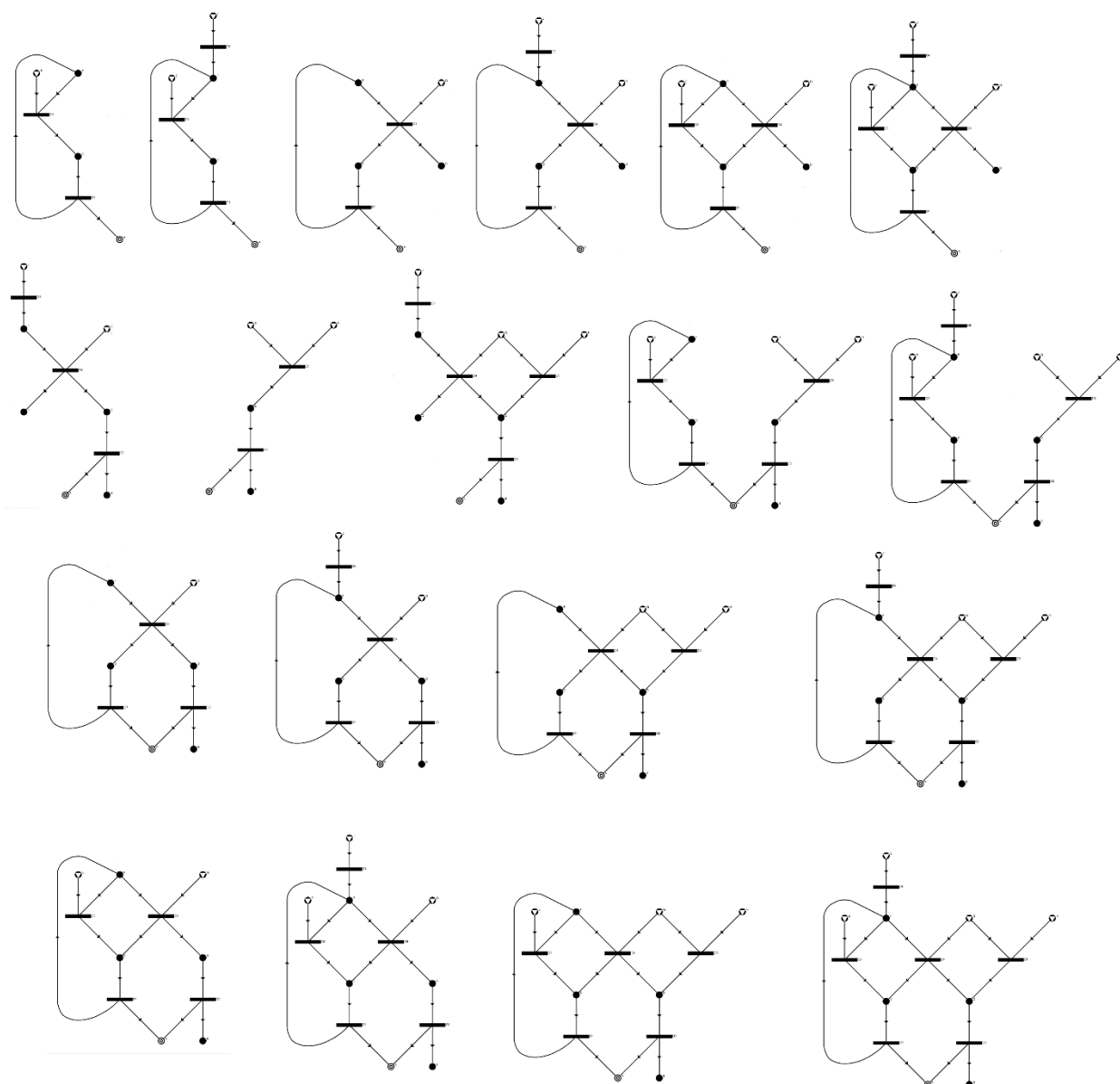
Ezekon felül a P-gráf keretrendszer tartalmaz több kombinatorikus algoritmust, amelyek közül a három leggyakrabban használt az MSG [12], az SSG [6], valamint az ABB [13]. Jelen munka során ezek közül az MSG és SSG algoritmusokat használja. Az MSG (Maximal Structure Generator) algoritmus célja a szintézis feladat maximális hálózatának meghatározása, vagyis kiszűri azokat a műveleti egységeket és anyagokat, amelyek miatt a hálózat nem teljesíti a korábban említett axiómákat.

Az SSG (Solution Structure Generator) algoritmus az MSG által meghatározott maximális struktúrából indul ki, és célja az összes lehetséges megoldás struktúra meghatározása. Szemléltető példának tekintsük az 1. ábrán látható, 6 műveleti egységes hálózatot. Erre a hálózatra az SSG algoritmus 19 megoldás struktúrát generál, amelyeket a 2. ábra sorol fel.



1. Ábra. 6 műveleti egységes szemléltető példa

Mivel a dolgozat egy későbbi fejezetében egy módosított SSG algoritmus mutat be, így itt szerepel az SSG algoritmus részletes, majd formális leírása. Az algoritmus rekurzív működésű, minden lépésben kiválaszt egy anyagot a gráfban, és dönt annak a gyártási lehetőségeiről, ezáltal több ágat létrehozva a keresési fában. Mivel a cél olyan hálózatok generálása, amelyek minden terméket képesek gyártani, ezért az algoritmus indulásakor tudja, hogy minden termék legyártásáról döntést kell hozni, így első lépésben az egyik termék gyártásáról dönt. Minden ág az adott anyag legyártásának egy lehetőségét jelöli a gyártásra kiválasztott műveleti egységek halmazával. Ezzel együtt nyilván ismeri kell a döntések során nem kiválasztott műveleti egységeket, hiszen ezek nem szerepelhetnek az adott ágban generált hálózatban, így később sem választhatóak be. Hasonlóan a már beválasztott műveleti egységek későbbi lépésben nem zárhatóak ki. A rekurzív meghívások során az algoritmus számon tartja, hogy melyik anyag csúcsokról kell még dönteni, hogy melyekről döntött már, valamint, hogy az adott lépésig eljutva melyik műveleti egységeket választotta be a hálózatba, és melyikeket zárta ki onnan. Egy anyagról született döntés után a kiválasztott műveleti egységek összes bemeneti anyaga is a legyártandó anyagok közé kerül, kivéve ami nyersanyag, vagy már döntöttünk róla.



2. Ábra. A szemléltető példa megoldás struktúrái

2.1 Az SSG algoritmus

Globális adatok: nyersanyagok halmaza (R), termékek halmaza (P), összes anyag halmaza (M), műveleti egységek halmaza (O)

Bemenet: döntésre váró anyagok halmaza (p), azon anyagok halmaza, amelyekhez már született döntés (m), az eddigiek során beválasztott (i), valamint kizárt (e) műveleti egységek halmaza

Megjegyzés: egy adott $x \in M$ anyag esetén $o^-(x)$ jelöli az anyagot legyártó műveleti egységeket, valamint egy adott $o \in O$ műveleti egység esetén $m^-(o)$ jelöli a műveleti egység bemeneti anyagait

Kimenet: megoldás struktúrák halmaza (S)

Az algoritmus kezdeti meghívása:

$$S = SSG(P, \emptyset, \emptyset, \emptyset)$$

Formális leírás:

$SSG(p, m, i, e)$

kezdés

$S = \emptyset$

ha $p = \emptyset$ **akkor**

visszaad S

elágazás vége

legyen $x \in p$ egy tetszőleges anyag

$C := \wp(o^-(x)) \setminus \emptyset$

minden $c \in C$ halmaz esetén

ha $c \cap e = \emptyset$ és $(o^-(x) \setminus c) \cap i = \emptyset$ **akkor**

$S = S \cup SSG(p \cup (\bigcup_{o \in c} m^-(o)) \setminus (R \cup m \cup \{x\}), m \cup \{x\}, i \cup c, e \cup (o^-(x) \setminus c))$

elágazás vége

ciklus vége

visszaad S

algoritmus vége

2.2 Teljes bejárásos algoritmus

Jelen dolgozat ugyan nem folyamat-hálózat szintézissel foglalkozik, hanem folyamatok megbízhatóságának meghatározásával, azonban látható lesz, hogy a szintézis algoritmusok által képviselt elvek és ötletek itt is hatékonyan felhasználhatóak. A folyamat megbízhatósága elsősorban kettő adat függvénye: (i) a folyamatban lévő műveleti egységek saját megbízhatóságai, és (ii) a műveleti egységek strukturális kapcsolatai. A számítások során feltesszük, hogy mind a műveleti egységek, mint a rendszer állapota bináris, vagyis vagy működőképes, vagy nem, illetve, hogy a műveleti egységek meghibásodásai egymástól függetlenek. Jelölje egy n műveleti egységet tartalmazó rendszerben az i műveleti egység megbízhatóságát p_i , $i = 1, 2, \dots, n$, ahol $0 \leq p_i \leq 1$. Következésképpen a műveleti egység meghibásodásának a valószínűsége $1 - p_i$.

A P-gráf alapú megbízhatóság-számító algoritmus [4] az egyszerű valószínűség-számítási elveken alapul. Lényege, hogy egy n műveleti egységet tartalmazó folyamat, vagy rendszer, a műveleti egységek állapotai alapján összesen 2^n állapottal rendelkezik, amelyek együttesen egy teljes esemény-rendszert alkotnak. Ebből a 2^n állapotból vannak olyanok, amelyekben a rendszer még működőképes a meghibásodások ellenére is, valamint vannak olyanok, amelyekben már nem. A korábbi módszer végig járja a 2^n állapotot, mindegyiknek megvizsgálja a működőképességét, a működőképeseket kigyűjti az U halmazba, majd ezek valószínűségeit összeadva határozza meg a folyamat megbízhatóságát.

Példának vegyük az 1. ábrán mutatott hálózatot, 6 műveleti egységgel (O1, O2, O3, O4, O5, O6). A 2^n állapot vizsgálata során az algoritmus mindegyik állapot által jelölt részhálózatra a P-gráf keretrendszerben ismert MSG algoritmust futtatja le. A részhálózat működőképes, amennyiben az MSG algoritmus talált maximális struktúrát a részhálózatban. Ez azt jelenti, hogy van még olyan része a hálózatnak, amely képes a termékeket legyártani. Amennyiben az MSG algoritmus szerint az adott részhálózatban

The graph consists of nodes A, B, C, D, E, F, G, and H. Node A is a source (circle with a dot), and node B is a sink (circle with a dot). Nodes C and D are intermediate nodes. Edges are labeled with flow values: O1, O2, O3, O4, O5, O6. The graph shows a complex network of paths and cycles, with some edges highlighted in grey.

The graph consists of nodes A, B, C, D, E, F, G, and H. Edges are labeled O1 through O6. Capacities are indicated by numbers on the edges: O1 (10), O2 (10), O3 (10), O4 (10), and O5 (10). A path is highlighted in gray, starting from A and ending at H, passing through D, G, and F.

10

$2^6 = 64$ részhálózat. Ezek mindegyikének ellenőrzése után az algoritmus összesen 36 működőképes részhálózatot talált.

Egy részhálózat tulajdonképpen egy elemi eseményt jelöl a részhálózatok által megadott teljes esemény rendszerből. Az $(x_1, x_2, \dots, x_n)$ vektorral leírt részhálózat valószínűsége a műveleti egységek meghibásodásainak függetlenségéből adódóan:

$$P((x_1, x_2, \dots, x_n)) = \prod_{i=1}^n p_i^{x_i} (1 - p_i)^{(1-x_i)}$$

Mivel a részhálózatok elemi eseményeknek felelnek meg, ha adott a működőképes részhálózatok U halmaza, akkor a folyamat megbízhatósága az U halmaz elemeinek összesített valószínűségével egyezik meg:

$$\sum_{(x_1, x_2, \dots, x_n) \in U} \prod_{i=1}^n p_i^{x_i} (1 - p_i)^{(1-x_i)}$$

2.3 Bináris Döntési Diagram

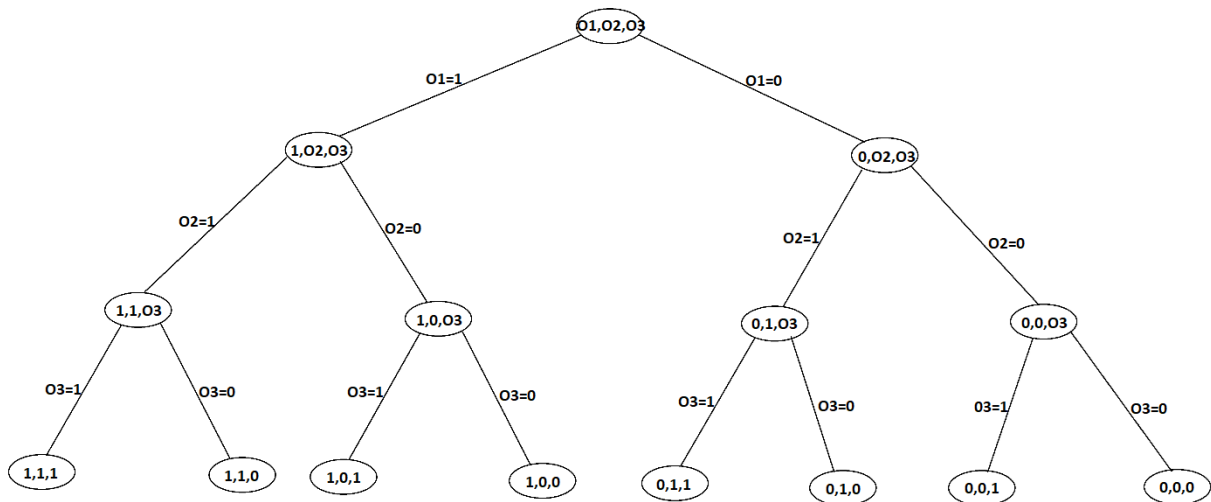
A leírt, nyers megbízhatóság számító alkalmazás mellett kiindulási alapként megvizsgálásra került a Bináris döntési diagram alapú gyorsítási eljárás is. A Bináris döntési diagram (Binary Decision Diagram – BDD) elve az, hogy a műveleti egységek állapotát bináris változóként kezelve a folyamata 2^n állapotát egy bináris fába rendezi, amelynek megfelelő bejárásával csökkenthető a működőképességi vizsgálatok (MSG algoritmus futtatásának) mennyisége. Ez a módszer a rendszerek koherens tulajdonságát aknázza ki. Egy rendszer koherens, ha a működőképességi függvény monoton, és a rendszer mindegyik műveleti egysége releváns a működőképesség szempontjából. Ez azt jelenti, hogy ha nem működik egyik műveleti egység sem akkor a rendszerünk sem fog működni és ha egy működőképes rendszert kiegészítünk egy működőképes műveleti egységgel akkor az nem fogja elrontani a működését. Ha adott kettő részhálózat, H_1 , és H_2 , ahol H_1 része H_2 -nek, akkor:

- ha H_1 működőképes, akkor H_2 is, valamint
- ha H_2 nem működőképes, akkor H_1 sem.

H_1 akkor része H_2 -nek, ha a H_1 -ben működőképes műveleti egységek halmaza részhalmaza a H_2 -ben működőképes műveleti egységek halmazának. Ezzel együtt az is teljesül, hogy a H_2 -ben működőképtelen műveleti egységek halmaza részhalmaza a H_1 -ben működőképtelen műveleti egységek halmazának.

A BDD során felépített bináris fa levelei a rendszer állapotai. A korábbi szinteken döntéseket végez a műveleti egységekhez tartozó bináris változókon. A bináris fa gyökere egy olyan csúcs, ahol még egyik műveleti egységről sem született döntés. Itt a módszer megvizsgálja, hogy a rendszer működőképes-e, amennyiben minden műveleti egység működőképes. Más szóval azt vizsgálja meg, hogy a csupa egyest tartalmazó bináris vektor által jelölt állapot működőképes-e. Az feltehető, hogy a csupa nullást tartalmazó vektor által jelölt állapot esetén, azaz amikor egyik műveleti egység sem működik, a rendszer nem működőképes.

A fa mindegyik csúcsához döntések sorozata vezet el, minden döntés egy műveleti egység állapotának meghatározásáról szól: a hozzá tartozó bináris változót vagy 0-ra, vagy 1-re állítja az algoritmus. Az 5. ábra mutat egy példát a teljes BDD-re 3 műveleti egység esetén. Minden csúcsban a 0 érték azt jelenti, hogy az a műveleti egység nem része a hálózatnak, az 1 azt jelenti, hogy a műveleti egység működőképes, a változó értékek (O1, O2, O3) pedig azt jelentik, hogy arról csak a későbbi lépésekben születik döntés.



5. Ábra. Bináris Döntési Diagram 3 változó esetén

Minden csúcsban két esetet kell vizsgálni, a csúcs alatti részében lévő legjobbat és legrosszabbat. A legjobb eset az, amikor minden, még el nem döntött változó értéke 1. Hasonlóan, a legrosszabb az, amikor mindegyik 0. Ezek az adott csúcs alatt lévő részfához tartozó levelek közül a legtöbb, vagy legkevesebb hibával rendelkező levelek. Például a 6 műveleti egységes példa esetén az (1, 0, O3, O4, O5, O6) csúcsból kiindulva a legjobb levél az (1,0,1,1,1,1), míg a legrosszabb levél az (1, 0, 0, 0, 0, 0). Ez a két csúcs azért érdekes, mert a rendszer koherens tulajdonsága alapján ezek korlátozzák alulról és felülről az összes többi esetben a rendszer állapotát. Ezen korlátok ismeretében külön vizsgálat nélkül lehet információt szerezni a részfához tartozó többi levélről is. A rendszer koherens tulajdonsága alapján következik, hogy

- Ha egy csúcs alatti részében a legjobb lehetséges esetben sem működőképes a rendszer, akkor egyiken sem lesz az.
- Ha egy csúcs alatti részében a legrosszabb lehetséges esetben is működőképes a rendszer, akkor mindegyikben az lesz.

Ha a kettő feltétel közül bármelyik teljesül, az azt jelenti, hogy a csúcsához tartozó részfa további megvizsgálására már nincsen szükség, hiszen már minden levél működőképesége ismert. Ez alapján a fa csúcsaiban vágás végezhető, és az állapotok ellenőrzései közül jelentős mennyiségű elhagyható.

A fa minden csúcsában 2 vizsgálat eredményét kell meghatározni, ami csúcsonként kettő működőképességi teszt lenne. A teljes fa vizsgálata során ez több vizsgálatot jelentene, mint az eredeti algoritmus esetén. Azonba ezen tesztek közt sok redundáns vizsgálat is megtörténne. A fában minden csúcsnak kettő gyereke van, az egyikben a következő bináris változó értéke 0 (nullás irány), a másikban 1 (egyes irány). Észrevehető, hogy a csúcs nullás irányhoz tartozó gyerekének alsó korlátja megegyezik az adott csúcs alsó korlátjával, és hasonlóan az egyes irányhoz tartozó gyerekének felső korlátja is megegyezik az adott csúcs felső korlátjával. Ebből következően a gyökeret kivéve minden csúcsra igaz, hogy a két meghatározandó korlát közül az egyik megegyezik a szülőnek ugyanezzel a korlátjával. Ennek megfelelően csak a gyökérben kell kettő vizsgálatot végezni (illetve egyet, hiszen a műveleti egység nélküli rendszer nem lesz működőképes), a fa többi köztes csúcsában elegendő egy vizsgálat is. Továbbá, a levelekben egy vizsgálatra sincs szükség, hiszen akkor már nincs több eldöntendő változó, és az eredmény a szülőből ismert. Ezeket figyelembe véve a BDD alapú számítás a legrosszabb esetben sem igényel több vizsgálatot, mint az eredeti algoritmus.

Természetesen a BDD alapú algoritmus akkor lesz igazán hatékony, ha nem csak a levelek működőképességét képes előre meghatározni, hanem képes ezen levelek vizsgálatának teljes elkerülésére. Ahhoz, hogy a rendszer megbízhatósága hatékonyan kiszámítható legyen BDD segítségével, a számítást igazítani kell, hogy a fa csúcsaiban végzett vágások során a kivágott működőképes levelek valószínűségeit is képes legyen figyelembe venni. A hagyományos számítás a fa működőképes hálózatokat adó leveleihez határozza meg a hozzájuk tartozó valószínűségeket majd ezeket adja össze. Azok a vágások melyek esetében a legjobb eset sem működőképes, ezt a számítást nem befolyásolják, hiszen azon az ágon egyik levél valószínűségét sem kell a megbízhatóságba beszámítani.

Azoknál a vágásoknál azonban, ahol a legrosszabb eset is működőképes, ott csúcsból induló részfában minden levél működőképes, ezek valószínűségeit hozzá kell adni a rendszer megbízhatóságához. Ez a számítás hatékonyan elvégezhető a következőkben leírt észrevételnek köszönhetően.

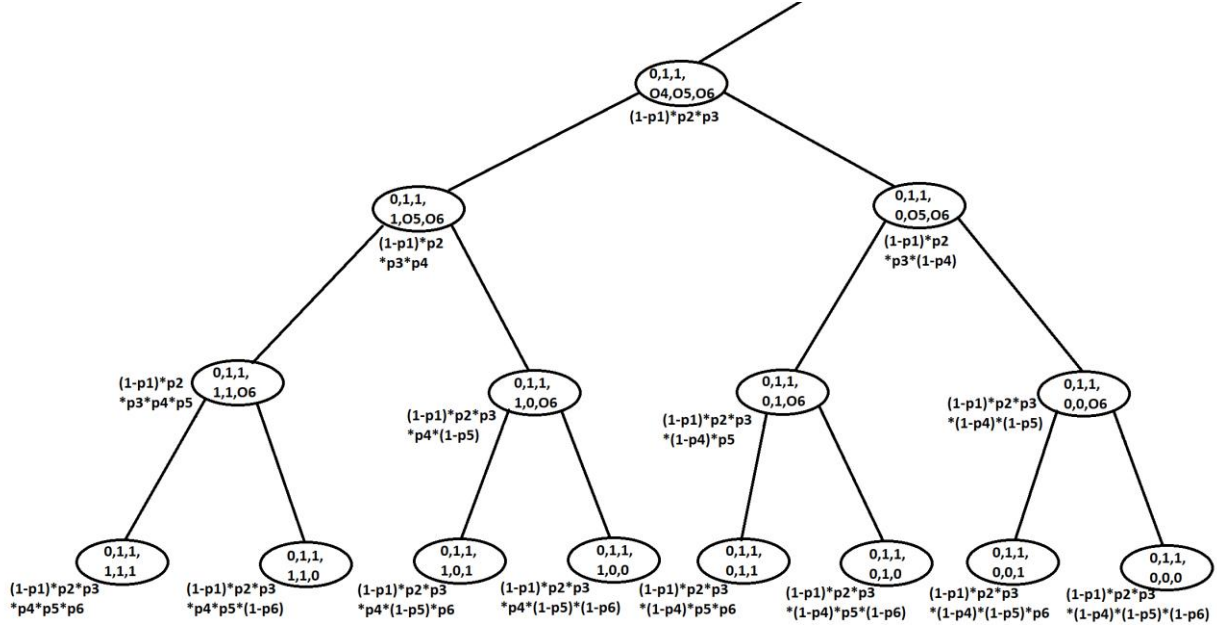
Amennyiben a csúcsban az első k érték adott, akkor még szabadon van $n - k$ bináris érték. Jelölje ezt a csúcsot a $(b_1, b_2, \dots, b_k, x_{k+1}, x_{k+2}, \dots, x_n)$ csúcs, ahol a $b_1, b_2, \dots, b_k$ a már eldöntött értékek az $x_1, x_2, \dots, x_n$ pedig a még eldöntendőek. A még eldöntetlen $n - k$ érték alapján csúcs alatti részfa 2^{n-k} levelet fog tartalmazni, ahol ezek a változó értékek az összes lehetséges kombinációt felveszik.

A levelek valószínűségeiben az első k változóhoz tartozó részek közösek, ezek kiemelhetők. A kiemelés után megmaradt részek egy $n - k$ változóból álló teljes esemény rendszert írnak le, ahol ezáltal a valószínűségek összege 1-et fog eredményül adni. Ebből kifolyólag a csúcshoz tartozó teljes részfa eredő valószínűsége a csúcshoz tartozó $b_1, b_2, \dots, b_k$ értékek alapján határozható meg.

$$P((b_1, b_2, \dots, b_k, x_{k+1}, x_{k+2}, \dots, x_n)) = \prod_{i=1}^k p_i^{b_i} (1 - p_i)^{(1-b_i)}$$

A 6. ábra mutat egy példát. Ezen a példán egy 6 műveleti egységből álló rendszer megbízhatóságának meghatározása során az aktuális csúcsban az első három változóról

már született döntés. Az ábrán az (0, 1, 1, 04, 05, 06) csúcsól kiderül, hogy a részfájában minden levél működőképes hálózatot jelöl (mivel az alsókorlátja, vagyis a (0,1,1,0,0,0) csúcs is működőképes), így itt a számítás során vágást lehet végezni. Az algoritmus nem vizsgálja tovább a további részeket, hanem meghatározza az eredő valószínűségét, ami $((1 - p_1) * p_2 * p_3)$, és ezt számolja bele a rendszer megbízhatóságába.



6. Ábra. Megbízhatósághoz szükséges valószínűségek vizsgálata Bináris Döntési Diagram esetén

Az ábra (0, 1, 1, 04, 05, 06) csúcs részfájában minden levél működik, tehát a megvizsgálásuk nélkül a csúcs eredő valószínűsége számítható bele a megbízhatóságba.

$$\begin{aligned}
 P((0,1,1,04,05,06)) &= \\
 &P((0,1,1,0,0,0)) + P((0,1,1,0,0,1)) + P((0,1,1,0,1,0)) + P((0,1,1,0,1,1)) \\
 &\quad + P((0,1,1,1,0,0)) + P((0,1,1,1,0,1)) + P((0,1,1,1,1,0)) + P((0,1,1,1,1,1)) \\
 &= \\
 &(1 - p_1) * p_2 * p_3 * (1 - p_4) * (1 - p_5) * (1 - p_6) + (1 - p_1) * p_2 * p_3 * (1 - p_4) \\
 &\quad * (1 - p_5) * p_6 + (1 - p_1) * p_2 * p_3 * (1 - p_4) * p_5 * (1 - p_6) + (1 - p_1) \\
 &\quad * p_2 * p_3 * (1 - p_4) * p_5 * p_6 + (1 - p_1) * p_2 * p_3 * p_4 * (1 - p_5) * (1 - p_6) \\
 &\quad + (1 - p_1) * p_2 * p_3 * p_4 * (1 - p_5) * p_6 + (1 - p_1) * p_2 * p_3 * p_4 * p_5 \\
 &\quad * (1 - p_6) + (1 - p_1) * p_2 * p_3 * p_4 * p_5 * p_6 = \\
 &(1 - p_1) * p_2 * p_3 \\
 &\quad * ((1 - p_4) * (1 - p_5) * (1 - p_6) + (1 - p_4) * (1 - p_5) * p_6 + (1 - p_4) * p_5 \\
 &\quad * (1 - p_6) + (1 - p_4) * p_5 * p_6 + p_4 * (1 - p_5) * (1 - p_6) + p_4 * (1 - p_5) \\
 &\quad * p_6 + p_4 * p_5 * (1 - p_6) + p_4 * p_5 * p_6) = \\
 &(1 - p_1) * p_2 * p_3 * 1 = (1 - p_1) * p_2 * p_3
 \end{aligned}$$

3 Tabulistás algoritmus

Az alap algoritmus képes meghatározni tetszőleges rendszer megbízhatóságát, azonban minden lehetséges állapot letesztelése nagy számítási igényt eredményez. Ehhez egy lehetséges gyorsításként, a Bináris Döntési Diagram használható, azonban a jelenlegi munka célja további gyorsítási lehetőségek kifejlesztése, és a hatékonyságuk vizsgálata. A munka során fontos követelmény, hogy a meghatározott eredményt megegyezzen a korábbival.

A korábbi algoritmusok vizsgálata során kiderült, hogy a számítási kapacitás jelentős részét az állapotok tesztelése teszi ki, amely az MSG algoritmus segítségével történik. Így egy természetes gyorsítási irány lehet a tesztelések számának csökkentése. Mivel egy nagy méretű rendszerben jellemzően a működésképtelen állapotok vannak többségben, így a nagyobb hatékonyság növekedés ezek korai felismerésétől várható. Az állapotokat megfelelő sorrendben vizsgálva, és a rendszer koherens tulajdonságát kihasználva bizonyos állapotok működésképtelensége tesztelés nélkül is eldönthető a korábban tesztelt állapotok eredményei alapján.

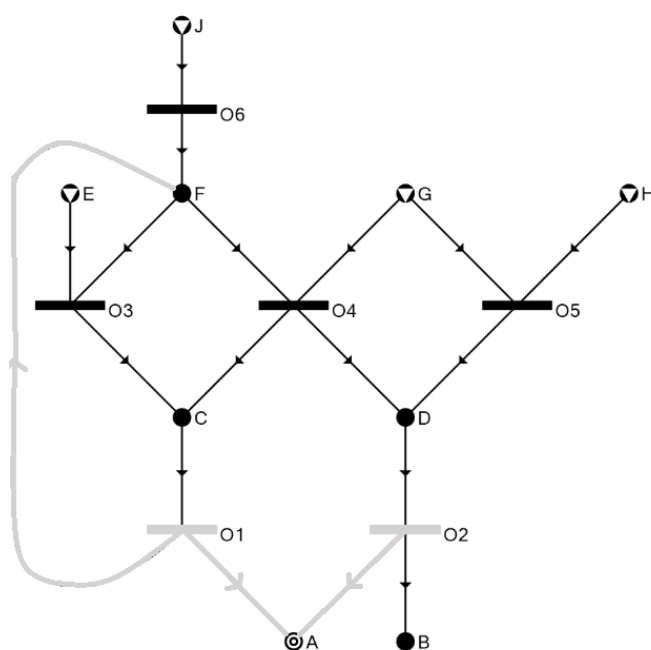
A dolgozatban vizsgált rendszerek koherens rendszerek, ennek következtében egy hálózat működő képességének vizsgálata után további következtetések vonhatóak le. Ha egy hálózat működőképes akkor minden olyan másik hálózat is működőképes, ahol legalább ugyan ezek a műveleti egységek működnek. Hasonlóan, ha egy hálózat nem működő képes akkor egyik olyan sem lesz működőképes, amelyben legalább ugyan ezek a meghibásodások történtek. Így, az algoritmus futtatása során, ha ismert, hogy egy részhálózat működésképtelen, akkor ebből több másik részhálózat működésképtelenségére lehet következtetni az MSG algoritmus futtatása nélkül is.

A gyorsítás megvalósításához az alapvető algoritmust ki kell egészíteni egy tabulistával, amely egy tároló a megvizsgált, és nem működőképesnek ítélt részhálózatok összegyűjtésére. Ha az MSG futtatás eredménye alapján a részhálózat nem működőképes, akkor a tesztelt eset elmentésre kerül ebben a tárolóban. Az ezután következő működőképeségi tesztek során, az MSG algoritmus futtatása előtt az algoritmus megvizsgálja, hogy a tesztelésre váró részhálózat része-e bármelyik már megtalált működésképtelen esetnek. Ha igen, akkor MSG futtatás nélkül kijelenthető, hogy az adott részhálózat nem működőképes.

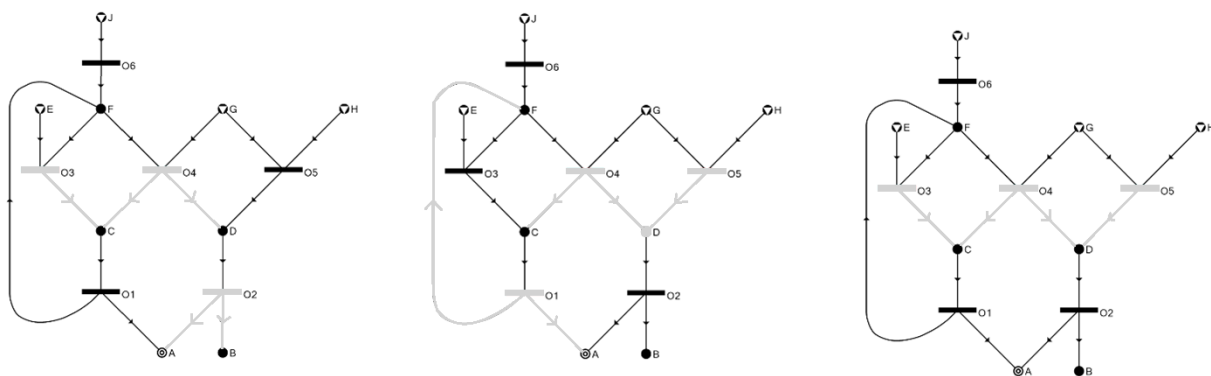
A lehető legnagyobb hatás érdekében olyan sorrendben kell végig járni a részhálózatokat, hogy ha egy részhálózat működésképtelensége a koherens tulajdonság alapján megállapítható akkor ez biztosan később legyen a vizsgálat során, mint az a részhálózat, ami alapján kizárható. Erre egy lehetőség, hogy a részhálózatok vizsgálati sorrendjében a meghibásodások száma monoton növekvő. Így az először megvizsgált részhálózat az, amelyikben nincs egy meghibásodás sem, aztán következnek azok, amelyekben 1 meghibásodás található, majd amelyekben 2, és így tovább.

Szemléltetésnek nézzük az 1. ábrán látható, 6 műveleti egységes P-gráfot. A bejárás abból az állapotból indul, ahol nincs egy működésképtelen műveleti egység sem. Ez a részhálózat értelemszerűen működőképes. Ezután következnek azok a részhálózatok,

Ezután következnek azon az állapotok, amelyekben kettő működésképtelen műveleti egység található. Az első működésképtelennek talált részhálózat: (0, 0, 1, 1, 1, 1), amely a 7. ábrán látható. Ebben az esetben az A terméket gyártó mindkettő műveleti egység meghibásodott, így a rendszer képtelen a további működésre, az algoritmus pedig elmenti a struktúrát, mint ismert működésképtelen állapot. A továbbiakban, ha egy struktúra O1 és O2 műveleti egysége hiányzik akkor ez alapján könnyen megállapítható, hogy nem működőképes. Ennek az egy működésképtelen részhálózatnak az ismerete a teljes algoritmus során 2^4 darab MSG hívással járó tesztelést spórol meg. A példa feladat esetében a többi 14 darab, kettő hibás műveleti egységet tartalmazó struktúra mindegyike működőképes.

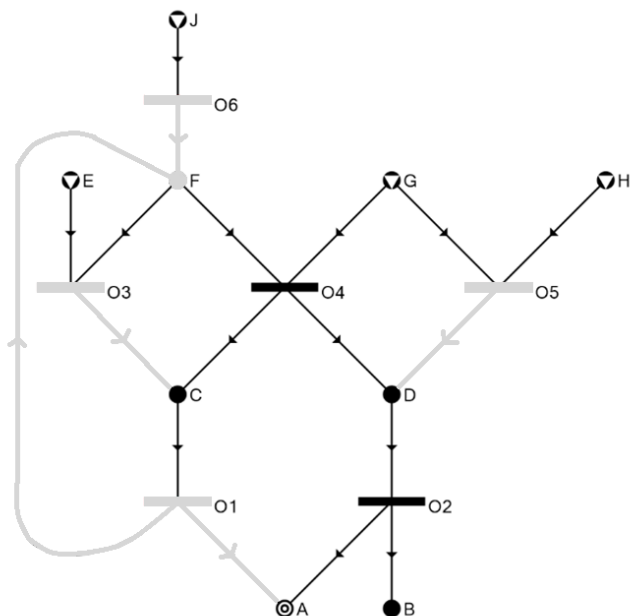


A következő körben azon struktúrák következnek, amelyek 3 meghibásodott műveleti egységet tartalmaznak. Ezen a szinten van 20 hálózat. Ezek közül 4 esetben, amelyekben az O1, O2 meghibásodott, már nem szükséges az MSG algoritmus futtatása a tesztelésre, mert a működésképtelenség a korábban elmentett részhálózat alapján megállapítható. A megmaradt 16 hálózatra szükséges az MSG algoritmus futtatása, ezek közül 13 működőképes, 3 pedig nem. Ez a három működésképtelen hálózat ((0, 1, 1, 0, 0, 1), (1, 0, 0, 0, 1, 1), (1, 1, 0, 0, 0, 1)) pedig bekerül a hibás hálózatok tárolójába, ezek a 8. ábrán láthatóak.



8. Ábra. A 3 meghibásodással rendelkező, új működésképtelen hálózatok

A további hálózatok vizsgálata során már csak egy olyan működésképtelen hálózatot talál az algoritmus, amelyre ez az eddigi hálózatok alapján nem határozható meg. Ez a hálózat, $(0, 1, 0, 1, 0, 0)$, amelyet a 9. ábra mutat, szintén bekerül a tárolóba. Ezzel együtt a példára összesen öt hibás hálózatot ment el az algoritmus a tárolóba.



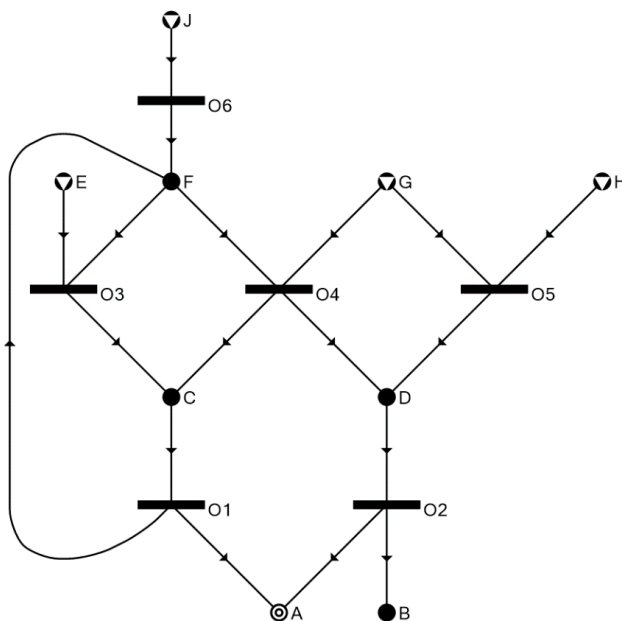
9. Ábra. Az utolsó működésképtelen hálózat, ami a tabulistába kerül

Az öt elmentésre került hálózat tehát: $(0, 0, 1, 1, 1, 1)$, $(0, 1, 1, 0, 0, 1)$, $(1, 0, 0, 0, 1, 1)$, $(1, 1, 0, 0, 0, 1)$ és a $(0, 1, 0, 1, 0, 0)$, amelyek segítségével az algoritmus a teljes futás során összesen 23 darab MSG hívást spórol meg. A 6 műveleti egységes rendszer esetén $2^6 (= 64)$ hálózatának ez közel 36%-a. Így az MSG algoritmus használatával történő működőképességi tesztet összesen csak 41-szer kell végrehajtani.

4 SSG (Solution Structure Generator for Reliability)

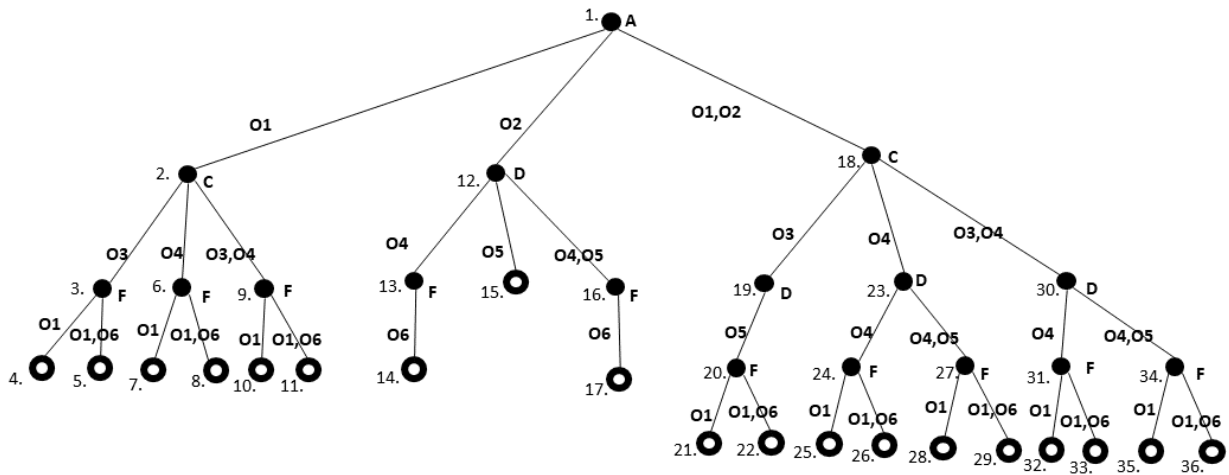
A Bináris Döntési Diagram működése során előkerült egy olyan észrevétel, hogy a sebességet nagyban befolyásolja, hogy melyik változóról dönt egy adott szinten az algoritmus. A következő gyorsítás elvét az az ötlet adta, hogy egy Bináris Döntési Diagram elvű algoritmus vélhetően hatékonyabb lenne, ha ki tudná használni a P-gráf alapú ábrázolásban tárolt extra strukturális információkat. A P-gráf keretrendszerben az SSG algoritmus az, amelyik a termékektől felépítkezve hatékonyan képes generálni az összes megoldás struktúrát, így az új algoritmus ennek a módosításával jött létre.

Az SSG algoritmus hatékonyságát az a tény biztosítja, hogy a meghozott döntések során nem generál olyan lehetőséget, amely zsákutcába visz. Az algoritmus által felépített keresési fában nincs szükség vágásra, minden egyes levél egy egyedi megoldás struktúrának felel meg. A 10. ábrán látható, 6 műveleti egységes szemléltető példára az SSG algoritmus a 11. ábrán látható keresési fát építi fel, amely 36 lépésből generálja a 19 megoldás struktúrát.



10. Ábra. A 6 műveleti egységes szemléltető példa

Mint a munka során kiderült, a rendszer megbízhatóságának meghatározására az SSG által felépített keresési fa majdnem megfelel. Minden egyes megoldás struktúra a megbízhatóság szempontjából is működőképes, valamint az is elmondható, hogy minden működőképes hálózat tartalmazza legalább az egyik megoldás struktúrát. Így tehát az új megbízhatóság meghatározó algoritmus azon az ötleten alapul, hogy az SSG algoritmust lehet-e úgy módosítani, hogy a megoldás struktúrák helyett a működőképes hálózatokat generálja hasonló hatékonysággal. Illetve még jobb az, ha nem is generálja az összes működőképes részhálózatot, hanem a Bináris Döntési Diagramhoz hasonlóan a keresési fa köztes csúcsaiban is képes vágást végezni.



11. Ábra. Az SSG algoritmus által bejárt keresési fa a szemléltető példára

Az algoritmus felépítéséhez azt is észre lehet venni, hogy az SSG algoritmus által generál megoldás struktúrák mindegyike definiál egy halmazt, amely működőképes részhálózatokat tartalmaz. Például a 11. ábrán látható keresési fa 10-es csúcsa azt a megoldás struktúrát jelöli, amely az O1, O3, O4 műveleti egységeket tartalmazza. Ezzel együtt azonban az ehhez a csúcshoz vezető döntések során az SSG algoritmus döntést hozott arról, hogy az O2 és O6 műveleti egységek nem kerülnek be a hálózatba. Az egyetlen műveleti egység, amiről nem született döntést az ehhez az csúcshoz vezető úton, az az O5. Így, a Bináris Döntési Diagramhoz hasonlóan a keresési fa ezen csúcsa felírható az $(1, 0, 1, 1, 0, 5, 0)$ vektorral, ami az 1 szabad változónak köszönhetően kettő működőképes hálózat együttesét jelenti, amelyek együttes valószínűsége a BDD-hez hasonlóan $p_1 * (1 - p_2) * p_3 * p_4 * (1 - p_6)$. Továbbá az is teljesül, hogy az SSG algoritmus csúcsai által jelölt halmazok diszjunktak, így az általuk meghatározott valószínűségek minden gond nélkül összeadhatóak.

Az előzőekben ismertetett elven az SSG algoritmus működése alapján meghatározható egy valószínűség, amely majdnem kiadja a rendszer megbízhatóságát. Azonban van egy hiányossága: a megoldás struktúrák által jelölt halmazok uniója nem tartalmazza az összes működőképes állapotot. Így tehát a kapott érték még nem a rendszer megbízhatósága. Azért, hogy a generálás teljes legyen, és ne maradjon ki működőképes struktúra a számításból, ki kell egészíteni az algoritmus működését. A módosított SSG algoritmus az eredetihez hasonlóan működik azzal az eltéréssel, hogy amikor egy anyag gyártásáról dönt, akkor megengedi azt is, hogy az adott anyag ne kerüljön legyártásra.

Az így keletkezett keresési fából már nem marad ki egyetlen működőképes hálózat sem, azonban már nem teljesül az az állítás, hogy csak működőképes hálózatokat generál az algoritmus. Ezért a keresési fa bizonyos csúcsai esetén szükség lehet a működőképesség tesztelésére a hagyományos, MSG algoritmust használó módszerrel: veszi a csúc alatti részfa legjobb esetét, vagyis a részben definiált hálózatban az összes, még el nem döntött műveleti egységet bevettnek tekinti, és lefuttatja az MSG algoritmust. Ha ez a teszt is működésképtelen hálózatot jelez, akkor a keresés adott irányában már nem létezik működőképes hálózat, így a keresést el lehet vágni. Minden olyan irány, ahol ez a tesztelés nem okoz vágást, a végén olyan levélhez vezet, ami működőképes struktúrák halmazát

jelöli. Továbbá igaz az, hogy az így kapott levelek továbbra is diszjunkt halmazokat jelölnek, amelyek uniója már tartalmaz minden működőképes hálózatot.

Annak érdekében, hogy ne kelljen sok felesleges működőképességi tesztet ellenőrizni, az algoritmus tovább javítható. A működőképességi tesztre a keresési fának csak azon ágaiban van szükség, amelyek eltérnek az SSG algoritmus eredeti működésétől. Ezek az ágak pedig pontosan azok, amelyekben a döntések sorozata tartalmaz olyan döntést, ahol a kiválasztott műveleti egység halmaza üres. Végezetül tehát a módosított SSG algoritmus leírása alább található, ami után egy példán keresztül szemléltetésre is kerül.

4.1 Az SSGR algoritmus

Globális adatok: nyersanyagok halmaza (R), termékek halmaza (P), összes anyag halmaza (M), műveleti egységek halmaza (O)

Bemenet: döntésre váró anyagok halmaza (p), azon anyagok halmaza, amelyekhez már született döntés (m), az eddigiek során beválasztott (i), valamint kizárt (e) műveleti egységek halmaza, továbbá egy logikai érték annak jelölésére, hogy az adott keresési ágon volt-e már olyan döntés, ami üres halmazt választott ki (t)

Megjegyzés: egy adott $x \in M$ anyag esetén $o^-(x)$ jelöli az anyagot legyártó műveleti egységeket, valamint egy adott $o \in O$ műveleti egység esetén $m^-(o)$ jelöli a műveleti egység bemeneti anyagait; egy $o \in O$ műveleti egység megbízhatóságát jelölje $r(o)$; a $TESZT(O)$ algoritmus ellenőrzi (az MSG segítségével), hogy az O halmazban lévő műveleti egységek által alkotott hálózat működőképes-e

Kimenet: a rendszer megbízhatósága (r)

Az algoritmus kezdeti meghívása:

$r = SSGR(P, \emptyset, \emptyset, \emptyset, hamis)$

Formális leírás:

$SSGR(p, m, i, e, t)$

kezdés

$r = 0$

ha $t = igaz$ **akkor**

ha $TESZT(O \setminus e) = hamis$ **akkor**

visszaad 0

elágazás vége

elágazás vége

ha $p = \emptyset$ **akkor**

visszaad $\prod_{o \in i} r(o) * \prod_{o \in e} (1 - r(o))$

elágazás vége

legyen $x \in p$ egy tetszőleges anyag

$C := \wp(o^-(x))$

minden $c \in C$ **halmaz esetén**

ha $c \cap e = \emptyset$ és $(o^-(x) \setminus c) \cap i = \emptyset$ **akkor**

ha $c = \emptyset$ **akkor**

$t' = igaz$

különben

$$t' = t$$

elágazás vége

$$r = r + SSGR(p \cup (\bigcup_{o \in c} m^-(o)) \setminus (R \cup m \cup \{x\}), m \cup \{x\}, i \cup c, e \cup (o^-(x) \setminus c), t')$$

elágazás vége

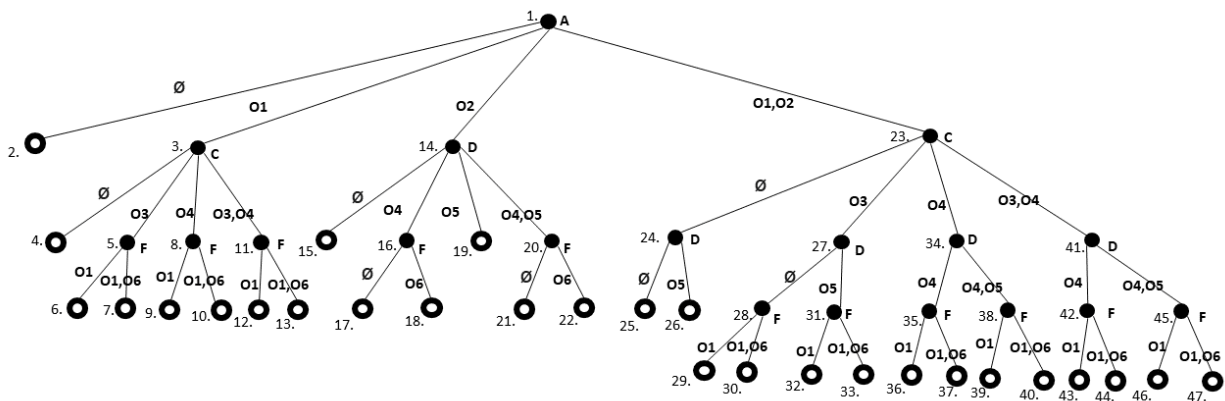
ciklus vége

visszaad S

algoritmus vége

4.2 Az algoritmus szemléltetése

A 10. ábra szemléltető példájára a módosított, SSGR nevet viselő algoritmus a 12. ábrán látható keresési fát határozza meg. Jól látható, hogy a hagyományos SSG algoritmushoz képest azon csúcsoknál, ahol az üres halmazt kiválasztó döntés nem mond ellent a korábbi választásokkal, úgy ezen döntéshez tartozóan is generálódik egy csúcs. A fában az üres körök azokat a csúcsokat jelöli, amelyekben a p halmaz üres, vagyis nem lehet már hozzá gyereket generálni. Amennyiben egy ilyen csúcs működőképes hálózatot reprezentál, úgy a hozzá tartozó valószínűséget az algoritmus beleszámolja a megbízhatóságba. A következőkben részletesen ismertetjük az algoritmus működését erre a példára.



12. Ábra. Az SSGR algoritmus által bejárt keresési fa a szemléltető példára

Az algoritmus az A anyag, azaz a termék gyártásának eldöntésével kezd. Ennek 4 gyereke van, hiszen az A terméket az O1 és O2 műveleti egységek képesek gyártani. Első lépésként megvizsgálja azt az esetet amikor kizárja az O1-et és az O2-t (2-es csúcs). Mivel ez egy üres halmazt kiválasztó döntés volt, ezért a működőképességet teszteli az MSG algoritmus futtatásával. A teszt eredménye alapján ebben az irányban nem található már működőképes hálózat. A következő vizsgált csúcsban (3-as) az O1 műveleti egység benne van a hálózatban, míg az O2 nincs. Ehhez a csúcshoz nem vezetett üres halmazos döntés, így ez megfelel az SSG eredeti működésének. Ennek megfelelően ebben az irányban biztosan lesznek működőképes csúcsok, nem szükséges a tesztelés futtatása. Az O1 műveleti egységnek 1 bemeneti anyaga van, a C, vagyis ennek a gyerekeit a C anyag legyártása alapján képezi. Itt ismét 4 lehetőség van, mivel kettő műveleti egység jöhet szóba: O3 és O4.

Mivel az algoritmus rekurzív, így mélységi bejárás szerint halad. A következő megvizsgált csúcs a 3-as csúcs első gyereke, a 4-es, amihez üres döntés vezet. A tesztelés futtatása tehát szükséges, ami megmondja, hogy ebben az irányban nem lehet működőképes hálózat. Az

5-ös csúcsban kiválasztásra kerül az O3 míg az O4 kizárva marad. Ez megfelel az SSG eredeti működésének, így tesztelés nincs. Az O3 működéséhez szükség van az E-re és az F-re. Az E egy nyersanyag, így ezt nem kell figyelembe venni a döntések során. Az F gyártásáról döntve, azt 2 műveleti egység gyártja: O1 és O6. Itt ismételt 4 lehetőség lenne, azonban, mivel az O1-es műveleti egység már szerepel a hálózatban, így a lehetőségek közül az a kettő, amelyekben az O1 nincs benne, ellent mond a korábbi döntéseknek. Valójában tehát csak 2 olyan lehetőség van, amely nem mond ellent a korábbi döntéseknek: csak az O1-et választjuk be, vagy mindkettőt. Mindkettő generált gyerek (6-es és 7-es csúcsok) megfelel az eredeti SSG működésének, és mivel a p halmaz mindkettőben üres, ezért ezek működőképes hálózatokat jelölnek. A 6-os csúcsban az i halmaz tartalma $\{O1, O3\}$, az e halmazé pedig $\{O2, O4, O6\}$, így a kiszámolt valószínűség: $p_1 * (1 - p_2) * p_3 * (1 - p_4) * (1 - p_6)$. Hasonlóan meghatározható a 7-es csúcsban kiszámolt valószínűség: $p_1 * (1 - p_2) * p_3 * (1 - p_4) * p_6$.

Az algoritmus működése a 8-as csúccsal folytatódik, amelyben az O4 szerepel a hálózatban, és az O3 nem. Az O4-nek két bemenete van: F és G. Utóbbi nem igényel vizsgálatot mivel nyersanyag, tehát a döntést az F alapján kell elvégezni. Az 5-ös csúcsához hasonlóan itt is két gyerek van, amely mindkettő végleges, működőképes hálózatot jelöl. A 11-es csúcsban az algoritmus beválasztja az O3-at és az O4-et is, kizárva csak az O2 marad. Az 5-öshöz és a 8-ashoz hasonlóan ennek a két gyereke is végleges, működőképes hálózat lesz.

A következő csúcs (14-es) a gyökér következő gyereke lesz, ahol az O1-es műveleti egység nem szerepel a hálózatban, az O2-es viszont igen. Az O2 műveleti egységhez a D anyag kell, ezért erről kell dönteni. A D-t gyártja az O4 és az O5. Az a gyerek (15-ös), ahol mindkettőt kizárja az algoritmus tesztelés igényel, és nem működőképes. A 16-os csúcs részhálózatában az O4-es műveleti egység szerepel, az O5-ös nem. Az F anyag gyártásáról kell dönteni. Az F-et az O1 és az O6 gyártja. Az O1-et egy korábbi döntés már kizárta, így a 4 lehetséges döntésből csak kettő konzisztens a korábbiakkal. Az O6 kizárása esetén (17-es csúcs) a tesztelés meghatározza, hogy a hálózat nem működőképes, míg az O6 bevétele esetén nincs szükség tesztelésre, működőképes végső hálózatot kapunk.

A 18-as csúcsban az O5 műveleti egység bemenetei a G és a H, melyek nyersanyagok így nem szükséges további vizsgálat. A 20-as csúcs kiértékelése a 16-ossal hasonlóan történik, azzal a különbséggel, hogy itt az üres halmazos döntés után (21-es csúcs), a kapott hálózat működőképes. Ez lezárja a keresési fa gyökeréből ezt az ágat. Következik a 23-as csúcs, ahol az O1 és az O2 is a hálózat része. A működésükhöz még szükség van a C és a D anyagokra, ebből a szétválasztás a C alapján végez döntést. Ezt az anyagot az O3 és O4 gyártja melyeket a 24-es csúcs kizár. Ebben a csúcsban így tesztet kell végezni, azonban a működőképességi teszt során kiderül, hogy ebben a keresési irányban még található működőképes részhálózat. Így a fa ezen ága tovább vizsgálható, de minden további csúcs esetén szükséges lesz a működőképességi teszt. A D legyártásáról döntve az O5 műveleti egység beválasztása a kérdés (mivel az O4-et már egy korábbi döntés kizárta). Az O5 kizárása esetén (25-ös csúcs) a hálózat nem működőképes, a bevétele esetén (26-os) csúcs viszont igen. A működőképességi tesztet a 26-os csúcsban is el kellett végezni a korábbi üres halmazos döntés miatt.

A keresés során az utolsó üres halmazos döntés a 27-es csúcs kiértékelése során található, ahol a D anyag legyártásáról kell dönteni. Az így keletkező 28-as csúcsban, és annak gyerekeiben szükséges a teszt elvégzése. A 28-as csúcsnak a teszt szerint még vannak működőképes gyerekei, így lehet folytatni a generálást. A 29-es és 30-as csúcsokat generálva a működőképességi tesztet mindkettőre el kell végezni a korábbi üres halmazos döntés alapján, és az eredmény szerint mindkettő működőképes hálózat lesz. Mivel a p halmaz mindkettőben üres, ezért ezek végső hálózatok is, és a hozzájuk tartozó valószínűség hozzáadható a megbízhatósághoz. A keresési fa későbbi lépései során már nincs több üres halmazos döntés, így a fa maradék csúcsai teljesen megfelelnek az SSG generálásának, így működőképességi teszt nélkül megfelelőnek nevezhetők.

Végeredményben az algoritmus 47 lépést végzett, azonban ezek közül csak 11 esetben volt szükség az MSG alapú működőképességi teszt elvégzésére (2, 4, 15, 17, 21, 24, 25, 26, 28, 29, 30 csúcsok). Ezek közül 5 esetben (2, 4, 15, 17, 25) lett az eredmény működésképtelen hálózat. Így a végső megbízhatóságot 23 csúcs valószínűségeinek az összegéből lehet meghatározni.

5 Teszt eredmények

Ebben a fejezetben az új algoritmusok tesztelése, kiértékelése, valamint összehasonlítása következik. A tesztelés 2 példán keresztül történt, amelyből az első a korábban is ismertetett szemléltető példa, a második pedig nagyobb méretű, összetett rendszer elemzése. A tesztek során vizsgált algoritmusok:

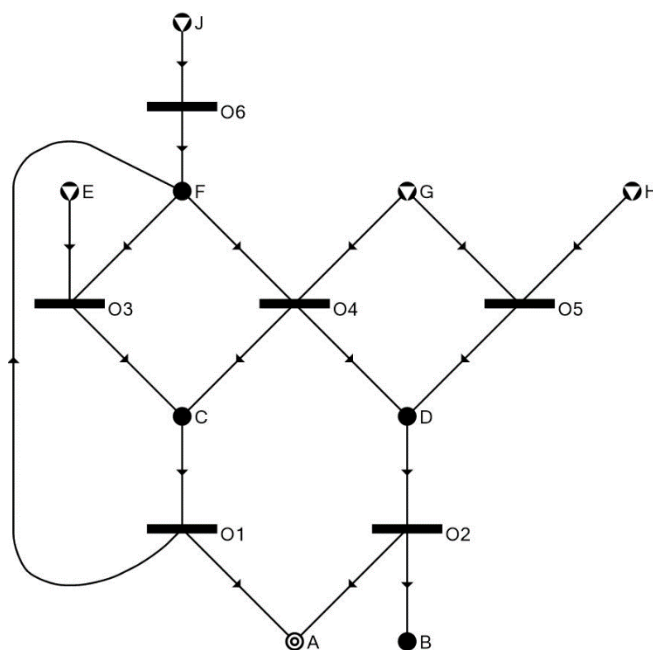
- Teljes bejárás
- Tabulistas algoritmus
- Bináris Döntési Diagram
- SSGR

A vizsgálat főbb szempontja:

- Futási idő
- MSG alapú működőképességi tesztek mennyisége

5.1 Szemléltető példa

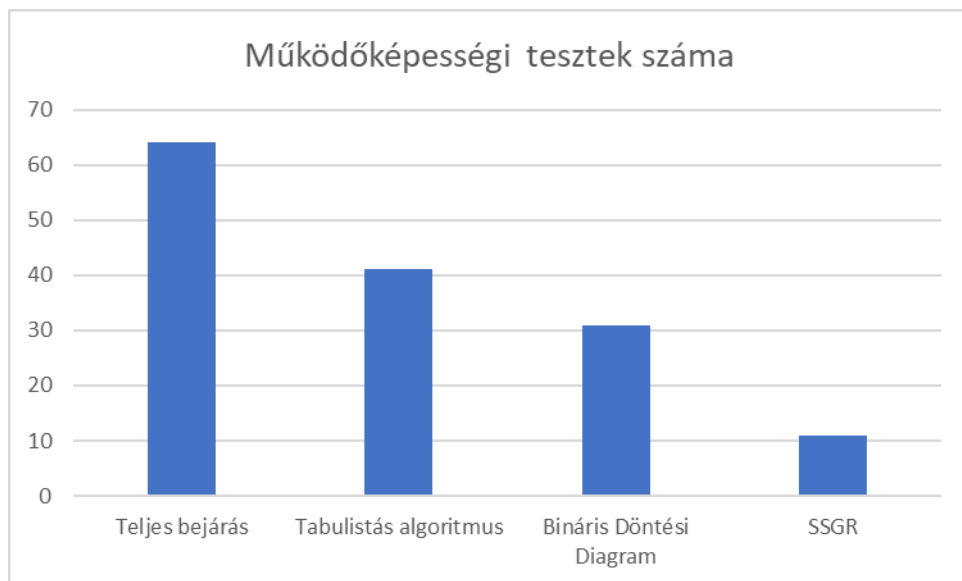
Az szemléltető példa hálózatát a 13. ábra ismételten tartalmazza az egyszerűség kedvéért. Ez a hálózat 6 műveleti egységből áll, 1 terméket állít elő (A) 4 nyersanyag felhasználásával (E, G, H, J).



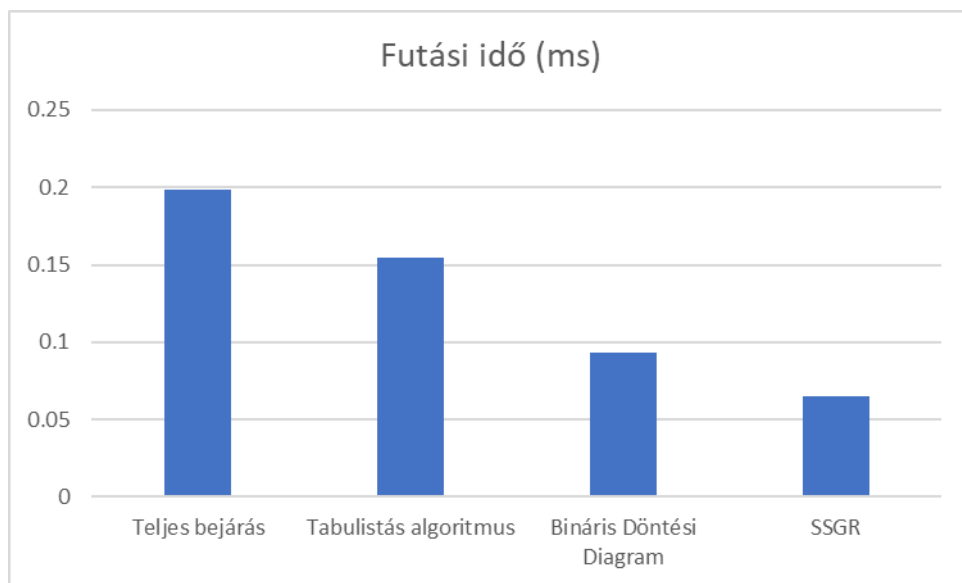
13. Ábra. A 6 műveleti egységes szemléltető példa hálózata

A példához tartozó futtatási eredményeket a 14. és 15. ábrák mutatják. Az ábrák egyértelmű kapcsolat mutatnak be az algoritmusok között. A teljes bejárás a leglassabb a 4 közül, hiszen mind a 64 állapotot tesztelnie kell. Ennél majdnem 25%-kal gyorsabb a tabulistas algoritmus, ennél is gyorsabb a Bináris Döntési Diagram, és a legjobb eredményt az SSGR algoritmus produkálta. A diagrammok alapján jól látható, hogy mint a futási idő, mind a tesztelések száma ugyanezt a sorrendet adja. Érdekes, hogy az SSGR algoritmus egyértelműen sokkal kevesebb tesztet végez, mint bármelyik másik. A futási idő tekintetében azonban csak kevéssel előzi meg a Bináris Döntési Diagramra épülő

módszert. Ennek az oka, hogy bár tesztelésből jóval kevesebbre van szüksége, az SSGR algoritmus további lépéseket is végez, és így a felépített keresési fa mérete hasonló lesz a BDD által bejárt fáéhoz.



14. Ábra. Működőképességi tesztek száma a szemléltető példára

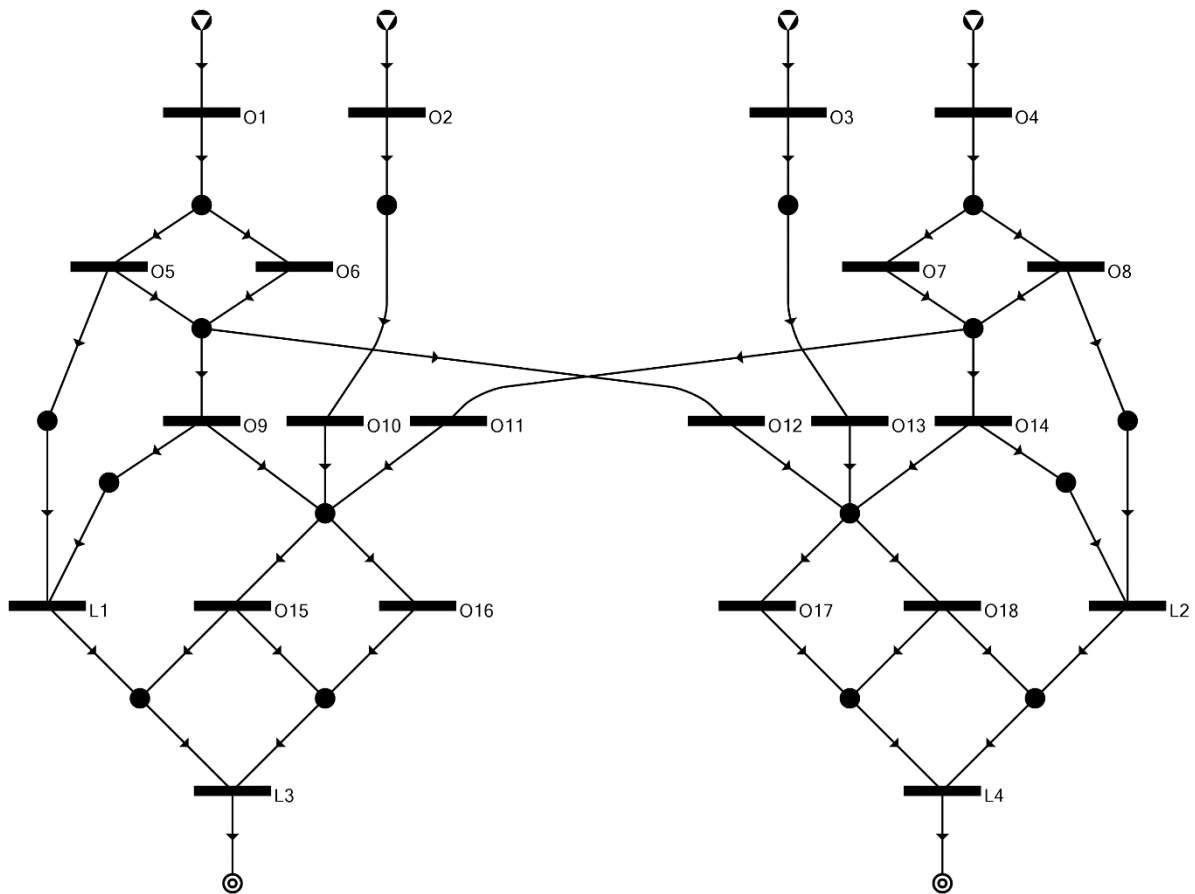


15. Ábra. Futási idők a szemléltető példára

5.2 Repülőgép üzemanyag rendszer

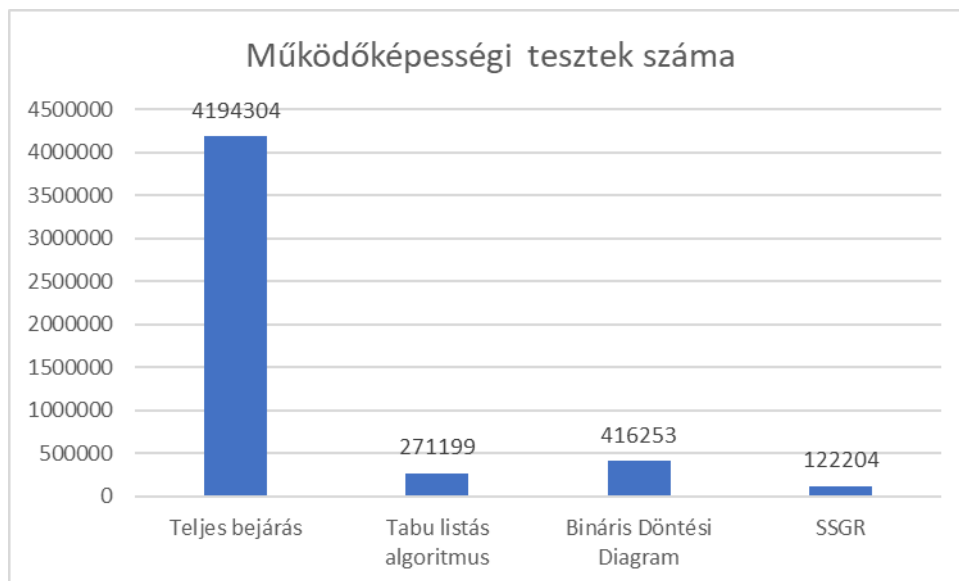
A második teszt példa egy repülőgép üzemanyag rendszerét modellezi. A modellezett repülő kettő motorral rendelkezik, amelyeknek a működését szimbolizálja a hálózat kettő

terméke. A 4 nyersanyag a két szárban lévő 2-2 üzemanyag tartályt jelöli. A rendszert leíró modell összesen 22 műveleti egységből áll, és a 16. ábrán látható.

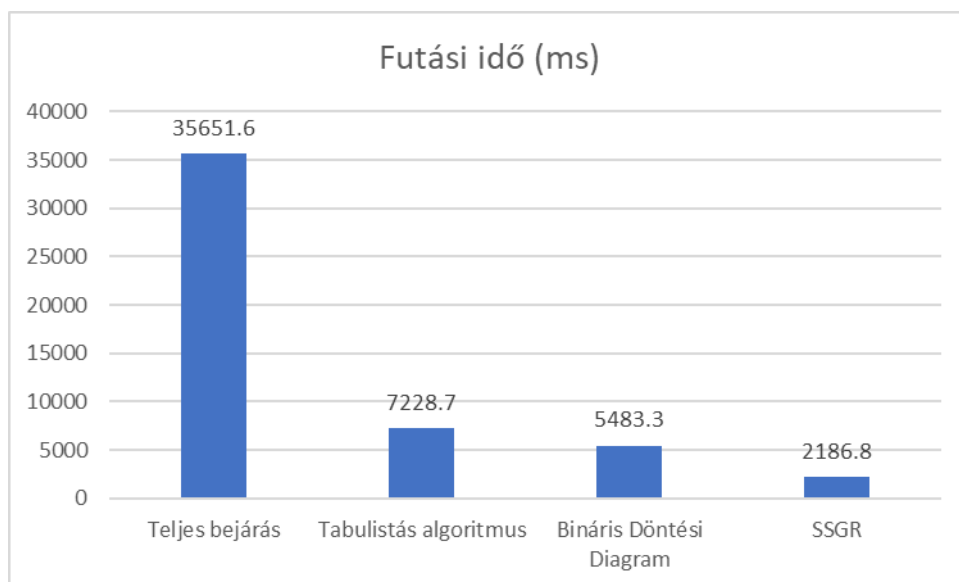


16. Ábra. 22 műveleti egységes példa, amely egy repülő üzemanyag rendszerét modellezi

A második példa eredményei a 17. és 18. ábrán láthatóak. Az előző példához képest erre a nagy hálózatra kicsit más eredmények jöttek ki. A gyorsítás nélküli, teljes bejárásos algoritmus továbbra is a leglassabb, ez esetben elég jelentősen. A leghatékonyabb algoritmus továbbra is az SSGR lett. A BDD és a tabulistas algoritmus kapcsolata viszont érdekes. A tabulistas algoritmus 35%-kal kevesebb működőképességi tesztet végez, mint a BDD algoritmus, viszont a futási mégis több, mint 30%-kal rosszabb. Ennek oka az, hogy bár a tabulistas algoritmus nem végzi el a teszteket az állapotok nagy részére, ettől még megvizsgálja a rendszer összes lehetséges állapotát. A nagy mennyiségű állapot végigjárása sok időt vesz igénybe, és habár a vizsgálatok nélkül még ez is jóval gyorsabb, mint a teljes bejárás, sebességben mégis alulmarad a kisebb keresési teret bejáró BDD-vel szemben.



17. Ábra. Működőképességi tesztek száma a második teszt feladatra



18. Ábra. Futási idők a második teszt feladatra

6 Összegzés

A dolgozat áttekintette a létező, P-gráf alapú megbízhatóság számító algoritmusokat. Ezután ezekre építve bevezetett kettő új megközelítést. Az egyik egy gyorsító eljárás a teljes bejárást végző algoritmushoz képest, ami továbbra is végig járja az összes állapotot, de egy tabulista segítségével jelentősen tudja csökkenteni az elvégzendő működőképességi tesztek számát. Ez jelentősen növeli a hatékonyságot, tekintve, hogy ezek a tesztek költséges műveletek.

A második bevezetett módszer egy új megközelítés, amely a P-gráf keretrendszer SSG algoritmusának működését veszi alapul, és azt egészíti ki a működőképes hálózatok meghatározására. Az új algoritmus a keresési tér vágásához a Bináris Döntési Diagram elvéből is merít ötletet, és ezzel minimalizálja a végrehajtandó működőképességi tesztek számát.

A dolgozat a két ismertetett módszer hatékonyságát példákon keresztül szemléltette, összehasonlítva azokat a meglévő módszerekkel. Elmondható, hogy már a tabulistas megközelítés is jelentős javulást ért el a teljes bejárásos módszerhez képest, viszont jellemzően alulmaradt a BDD algoritmussal szemben. A vizsgált tesz példákra a leghatékonyabb megközelítésnek az SSGR algoritmus bizonyult.

A jelenleg ismertetett algoritmusokban még sok a továbbfejlesztési lehetőség. A további kutatások során terven van az SSGR algoritmus továbbfejlesztése, valamint a BDD algoritmus fejlesztési lehetőségeinek vizsgálata is.

7 Irodalomjegyzék

- [1] Kumamoto H, Tanaka K, Inoue K (1977) Efficient evaluation of system reliability by Monte Carlo method. *IEEE T Reliab* 26(5):311-315
- [2] Prowell SJ, Poore JH (2004) Computing system reliability using Markov chain usage models. *J Syst Software* 73(2):219-225
- [3] Birolini A (2017) *Reliability Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg
- [4] Kovacs Z, Orosz A, Friedler F (2019) Synthesis algorithms for the reliability analysis of processing systems. *Cent Eur J Oper Res* 27(2):573-595
- [5] Chen Y, Ye Y, Yuan Z, Grossmann IE, Chen B (2022) Integrating stochastic programming and reliability in the optimal synthesis of chemical processes. *Comput Chem Eng* 157:107616
- [6] Friedler F, Tarjan K, Huang YW, Fan LT (1992) Combinatorial algorithms for process synthesis. *Comput Chem Eng* 16:S313-S320
- [7] Yenkie KM, Pimentel J, Orosz Á et al (2021) The P-graph approach for systematic synthesis of wastewater treatment networks. *AIChE J*:e17253
- [8] Kim Y, Park S (2006) Supply network modeling using process graph theory: a framework for analysis. *SICE-ICASE Int Jt Conf* 2006:1730-1733
- [9] Friedler F, Barany M, Bertók B et al (2011) Solving vehicle assignment problems by process network synthesis to minimize cost and environmental impact of transportation. *Clean Techn Environ Policy* 13:637-642
- [10] García-Ojeda JC, Bertók B, Friedler F (2012) Planning evacuation routes with the P-graph framework. *Chem Eng Trans* 29:1531-1536
- [11] Baumgartner J, Süle Z, Bertók B, Abonyi J (2019) Test-sequence optimisation by survival analysis. *Cent Eur J Oper Res* 27:357-375
- [12] Friedler F, Tarján K, Huang YW, Fan LT (1993) Graph-theoretic approach to process synthesis: Polynomial algorithm for maximal structure generation. *Comput Chem Eng* 17:929-942
- [13] Friedler F, Varga JB, Fehér E, Fan LT (1996) Combinatorially accelerated branch-and-bound method for solving the MIP model of process network synthesis. In: Floudas CA, Pardalos PM (eds) *Nonconvex optimization and its applications*. Kluwer Academic Publishers, Norwell, MA, pp 609-626