ALGORITMUSOK KIDOLGOZÁSA LOGIKAI, IDŐ ÉS MEGBÍZHATÓSÁGI FELTÉTELÉKET TELJESÍTŐ FOLYAMATHÁLÓZATOK SZINTÉZISÉHEZ

DOKTORI (PhD) ÉRTEKEZÉS

Kalauz Károly
témavezető: Dr. Bertók Botond

DOI: 10.18136/PE.2014.549

Pannon Egyetem
Műszaki Informatikai Kar
Informatikai Tudományok Doktori Iskola

2014

ALGORITMUSOK KIDOLGOZÁSA LOGIKAI, IDŐ ÉS MEGBÍZHATÓSÁGI FELTÉTELÉKET TELJESÍTŐ FOLYAMATHÁLÓZATOK SZINTÉZISÉHEZ

Értekezés doktori (PhD) fokozat elnyerése érdekében

Írta: Kalauz Károly

Készült a Pannon Egyetem Informatikai Tudományok Doktori Iskolája keretében

Témavezető: Dr. Bertók Botond

Elfogadásra javaslom (igen / nem)

(aláírás)

A jelölt a doktori szigorlaton .....%-ot ért el

Veszprém .....

a Szigorlati Bizottság elnöke

Az értekezést bírálóként elfogadásra javaslom:

Bíráló neve: ..... (igen / nem)

(aláírás)

Bíráló neve: ..... (igen / nem)

(aláírás)

A jelölt az értekezés nyilvános vitáján .....%-ot ért el

Veszprém .....

a Bíráló Bizottság elnöke

A doktori (PhD) oklevél minősítése .....

.....

Az EDT elnöke

Tartalomjegyzék

Tartalomjegyzék iii

Táblázatok jegyzéke vi

Ábrák jegyzéke vii

Kivonat x

Abstract xii

Abstrakt xiv

Köszönetnyilvánítás xvi

1. Bevezetés 1
- 1.1. Célkitűzések . . . 1
- 1.2. Saját eredményeim kiemelése . . . 3
- 1.3. Jelöléstan . . . 3

2. Irodalmi áttekintés 4
- 2.1. Folyamatok logikai struktúráját leíró módszerek . . . 4
  - 2.1.1. Üzleti folyamat modellezés . . . 4
  - 2.1.2. Petri háló . . . 7
  - 2.1.3. Állapot-tevékenység hálózat (STN) . . . 16
  - 2.1.4. Erőforrás-tevékenység hálózat (RTN) . . . 18
  - 2.1.5. S-gráf . . . 20
- 2.2. A folyamathálózat-szintézis kombinatorikus alapjai . . . 24
  - 2.2.1. A P-gráf . . . 29
  - 2.2.2. Strukturális tulajdonságokat leíró leképezések . . . 33
  - 2.2.3. A kombinatorikusan lehetséges struktúrák szükséges axiómái . 34
  - 2.2.4. Az MSG algoritmus . . . 35
  - 2.2.5. Az SSG algoritmus . . . 37
  - 2.2.6. Az ABB algoritmus . . . 39

3. PNS feladat fix részt tartalmazó lineáris költség függvénnyel **41**
3.1. A relaxált feladat 41
3.2. Modell paraméterek 42
3.2.1. Kiegészítő jelölések 43
3.3. Feladat meghatározás 44
3.4. Az ABB algoritmus paraméteres PNS feladatként történő leírása . . . 44

4. A PNS egymást kölcsönösen kizáró tevékenységek kezelésére **51**
4.1. Feladat leírás 51
4.2. Az SSGMX algoritmus 52
4.2.1. Az SSGMX algoritmus helyességének bizonyítása 55
4.3. A MILP modell 56
4.4. Az ABBMX algoritmus 57
4.4.1. Az ABBMX algoritmus helyességének bizonyítása 61
4.5. A kölcsönös kizárás kezelését szemléltető jármű hozzárendelési példa . 63
4.6. További feladatok futtatási eredményei 67
4.7. Összefoglalás 68
4.8. Kapcsolódó publikációim 69

5. A redundáns folyamat-hálózatok szerepe és fontossága **70**
5.1. A redundáns folyamat-hálózatok generálásának motivációja 70
5.2. A redundancia szerepe és fontossága 76
5.3. Redundancia generálás 81
5.4. Megbízhatóság meghatározása a P-gráf módszertanban 85
5.5. A redundancia és a megbízhatóság kapcsolata 88
5.6. Becslési eljárás megbízhatóság számításra 90
5.7. Tetszőleges struktúrák megbízhatóságának meghatározása 93
5.8. Szemléltető példa 99
5.9. Összefoglalás 101
5.10. Kapcsolódó publikációim 103

6. Időkorlátos PNS (TCPNS) **109**
6.1. A MILP modell 110
6.2. Az időkorlátok relaxációja 112
6.3. Szemléltető példa 113
6.4. Strukturális kiegészítés az előidejűség kezelésére 117
6.5. Az előidejűség kezelését szemléltető példa 120
6.6. A TCPNS-ben rejlő továbblépési lehetőség 127
6.7. Összefoglalás 128
6.8. Kapcsolódó publikációim 129

7. Összefoglalás **130**

8. Saját publikációk **132**

9. Új tudományos eredmények **136**

Irodalomjegyzék **138**

Tárgymutató **150**

Melléklet **151**
1. Melléklet 152
2. Melléklet 161
3. Melléklet 167

Táblázatok jegyzéke

2.1. A PNS feladat és a Petri háló elérhetőségi feladata közötti leképezés . 14

4.1. A jármű hozzárendelési tesztfeladat eredményéül kapott hozzárendelések és a megoldás során generált részproblémák száma 66
4.2. Futtatási eredmények kiértékelése 68

5.1. A megbízhatósági korlátra adott becslési eljárást szemléltető példa eredményei 93
5.2. Tetszőleges struktúra megbízhatóságának kiszámítása 96
5.3. A redundánsan kibővített struktúra megbízhatóságának kiszámítása . 104
5.4. A tevékenységek előfeltételei és következményei 105
5.5. Az alternatív ellátási szcenáriók teljes megbízhatósága 106
5.6. Az alternatív ellátási szcenáriók teljes megbízhatóságának kiszámítási formulái 107

6.1. Az entitások paraméterei 114
6.2. A tevékenységek előfeltételei, következményei és idő paraméterei . . . 116
6.3. Az entitások paraméterei 121
6.4. A tevékenységek előfeltételei, következményei és idő paraméterei . . . 122
6.5. A különböző módszerek összehasonlítása 128

Ábrák jegyzéke

2.1. Az üzleti folyamat modellezés főbb építő elemei 6
2.2. Az üzleti folyamat diagram, azaz BPD bemutatására szolgáló példa . 7
2.3. A petri hálót szemléltető példa 9
2.4. Az állapot-tevékenység hálózat, azaz az STN leírás elemei 16
2.5. Az állapot-tevékenység hálózat, azaz az STN leírást szemléltető példa 17
2.6. Az erőforrás-tevékenység hálózat, azaz az RTN leírás építő elemei . . 19
2.7. Az erőforrás-tevékenység hálózat, azaz az RTN leírást szemléltető példa 19
2.8. Az S-gráf leírás építő elemei 23
2.9. Az S-gráf leírást szemléltető példa 24
2.10. A hagyományos algoritmikus módszerek főbb lépései 27
2.11. A Keresési tér csökkentése 28
2.12. A P-gráf módszertan főbb lépései 30
2.13. Egy termelési feladat P-gráfja 31
2.14. Az MSG algoritmus 36
2.15. Az SSG algoritmus 38

3.1. Az ABB algoritmus 45
3.2. Az UpdateBound függvény algoritmusa 46
3.3. A RemoveSubproblems függvény 46
3.4. A SelectProblemABB függvény algoritmusa 47
3.5. A BranchABB függvény algoritmusa 48
3.6. Az NX algoritmus 49
3.7. Az RSG algoritmus 49
3.8. Az SolveRelaxedLP függvény algoritmusa 50

4.1. Az SSGMX függvény algoritmusa 53
4.2. A BranchSSGMX függvény algoritmusa 54
4.3. Az ABBMX függvény algoritmusa 57
4.4. A BranchABBMX függvény algoritmusa 59
4.5. A SolveRelaxedLPMX függvény algoritmusa 60
4.6. Az MX függvény algoritmusa 61
4.7. A kölcsönös kizárás kezelését szemléltető jármű hozzárendelési probléma maximális struktúrája 65

5.1. Egy IT szolgáltatás folyamatát bemutató példa 71
5.2. Az alap motivációs példa 72
5.3. A motivációs példa redundanciával kibővített változata 74
5.4. A logikai „ÉS” kapcsolat P-gráf reprezentációja 77
5.5. A logikai „VAGY” kapcsolat P-gráf reprezentációja 77
5.6. Két tevékenység együttműködésének alternatív strukturális reprezentációi 78
5.7. Két párhuzamos tevékenység alternatív szcenárióinak szuperstruktúrája 80
5.8. A redundancia generáló RG algoritmus 83
5.9. Az RG algoritmus szerepének szemléltetése 85
5.10. Entitás minőségi értékének meghatározása két bemenő él esetén . . . 87
5.11. Tevékenységen áthaladó információ minőségének számítása egy bemenő él esetén 88
5.12. Két tevékenység együttműködésének alternatív strukturális reprezentációi és azok megbízhatóságai 89
5.13. Két párhuzamos tevékenység alternatív szcenárióinak szuperstruktúrája 91
5.14. A megbízhatósági korlátra adott becslési eljárást szemléltető példa . . 92
5.15. Az irányítatlan kört tartalmazó szemléltető példa maximális struktúrája 94
5.16. Az irányítatlan kört tartalmazó szemléltető példa lehetséges megoldás struktúrái 95
5.17. Az irányítatlan kört tartalmazó szemléltető példa redundánsan kiegészített maximális struktúrája 97

5.18. A redundánsan kiegészített irányítatlan kört tartalmazó szemléltető példa lehetséges megoldás struktúrái 98
5.19. A szemléltető példa maximális struktúrájának P-gráf reprezentációja 100
5.20. A szemléltető példa maximális struktúrájának redundánsan kibővített P-gráf reprezentációja 102
5.21. A legmagasabb megbízhatósággal rendelkező megoldás struktúra P-gráf reprezentációja 108

6.1. Az időkorlátokkal bővített jármű hozzárendelési feladat maximális struktúrája 115
6.2. Az időkorlátokkal kibővített jármű hozzárendelési feladat megoldás struktúrája 118
6.3. Maximális struktúra előidejűség szerinti kiterjesztése az időkorlátok kezelésére 119
6.4. Az MSETC algoritmus 119
6.5. A szemléltető példa maximális struktúrájának előidejűség szerinti kiterjesztése az időkorlátok kezelésére 121
6.6. A feladat megoldásához vezető döntési fa 123
6.7. A megoldásstruktúrához vezető döntések szemléltetése 124
6.8. A döntési fa 5. állapotában az időkorlát túllépésének szemléltetése . . 125
6.9. Az előidejűséggel bővített időkorlát kezelést szemléltető példa megoldás struktúrája 126

Kivonat

Algoritmusok kidolgozása logikai, idő és megbízhatósági feltételeket teljesítő folyamathálózatok szintéziséhez

A vállalatok versenyképességének megőrzése érdekében elengedhetetlen azok üzleti folyamatainak tervezése és menedzselése. Amennyiben több döntési alternatíva létezik, akkor ezek értékelésében segítséget nyújthatnak a gyártási, logisztikai és üzleti folyamataik analízisét és optimalizálását támogató számítógépes megoldások. Ezzel szemben nem létezik olyan piacra bocsátott szoftver, amely támogatná a gyártási, logisztikai és üzleti folyamataik strukturális alternatíváinak generálását és azok felderítését.

Korábbi vizsgálatok jól tükrözik, hogy a folyamat-hálózat szintézisre (PNS) alkalmazott P-gráf módszertan, amelyet eredetileg vegyipari folyamatok modellezésére és optimalizálására dolgoztak ki, megfelelő eszköztárat biztosít a gyártási, logisztikai és üzleti folyamataik strukturális alternatíváinak előállítására és elemzésre. Az eredeti keretrendszer kiterjesztése a gyártási, logisztikai és üzleti folyamatokra jellemző megkötések kezelésére javíthatja a javasolt módszertan gyakorlati alkalmazhatóságát.

Kutatásom célja egy olyan módszertan kidolgozása, amely megfelelő alapot biztosít ellátási láncok (gyártási, logisztikai és üzleti folyamatok) matematikai és formális leírására és modellezésére, valamint az alternatív és optimális szcenáriók algoritmikus szintézisére, miközben figyelembe veszi a kölcsönös kizárásokat, határidőket és a strukturális redundanciát. A bemutatásra kerülő módszer az algoritmikus folyamatszintézis kombinatorikus alapelveire, speciálisan a P-gráf módszertanra épül.

Számos gyakorlati szempont definiálható a tevékenység halmazok kölcsönös kizárási megkötéseinek formájában. Némely megkötések segíthetnek csökkenteni a keresési teret, ezzel felgyorsítva magát a keresést. Ennek eredményeképpen sokkal jobban definiált alternatív struktúra halmazok állhatnak elő rövidebb számítási idő alatt.

Egy újonnan bevezetett algoritmus redundanciák beépítését definiálja alternatívaként egy folyamatszintézis feladat kiindulási struktúrájába, miután a folyamatszintézis algoritmusok változtatás nélkül futtathatóak, eredményül pedig redundanciával növelt és nem növelt hálózatokat is adnak eredményül.

A határidők betartása elengedhetetlen szempont a üzleti folyamtok és ellátási láncok tervezésében. A jelen munka integrálja az erőforrások és tevékenységek rendelkezésre állásának és a végcélok határidejének időkorlátait a matematikai modellbe, valamint a PNS módszertan algoritmusaiba.

Abstract

Development of algorithms for the synthesis of process networks performing logical, time and reliability criterion extension

Design and management of manufactural, logistic, and business processes are key factors for companies to be competitive. Available computer aid for manufactural, logistic, and business process analysis and optimization can help evaluating decision alternatives if available. In contrast, no software in the market supports exploring or generating structural alternatives for manufactural, logistic, and business processes.

Former examinations show that the P-graph approach to process-network synthesis (PNS) originally conceived for conceptual design of chemical processes provides appropriate tools for generating and analyzing structural alternatives for manufactural, logistic, and business processes. However, extension of the original framework to handle constraints specific to manufactural, logistic, and business processes may improve the practical applicability of the proposed methodology.

The aim of my research is to develop a methodology that provides an adequate basis to portray and model supply chains mathematically and formally as well as to synthesize optimal and alternative supply scenarios algorithmically while taking into account mutual exclusions, deadlines, and structural redundancy. The proposed methodology is based on the combinatorial foundations of algorithmic process synthesis or more specifically on the P-graph framework.

Numerous practical considerations can be defined in the form of constraints expressing mutual exclusiveness of sets of activities. Such constraint can help reducing the search space as well as accelerating the search. As a result practically more

specified set of alternative structures can be generated within a shorter computation time.

A novel algorithm incorporates potential redundancies into the initial structure of a PNS problem in a way that the original PNS algorithms can result networks with and without redundancy.

To satisfy the deadline is a crucial aspect in manufactural, logistic, and business process and supply chain design. In the present work time constraints on the availability of the resources, activities, and deadlines for the final targets are incorporated to the mathematical model, as well as, to solution algorithms of PNS.

Abstrakt

Entwicklung eines Algorythmus für die Synthesis von Prozessnetzwerken, der logische, zeitliche und funktionale Merkmale als Erweitwerung ausführt

Das Design und Management von manufactural, Logistik und Geschäftsprozessen ist ein Schlüsselfaktor für die Wettbewerbsfähigkeit von Firmen. Bestehende Computerprogramme können dabei helfen, bestehende Entscheidungsalternativen zu bewerten. Im Gegensatz dazu ist derzeit keine Software auf dem Markt, die strukturelle Alternativen für manufactural, Logistik und Geschäftsprozesse untersucht oder generiert.

Voruntersuchungen haben gezeigt, dass die P-Graph Methode der Prozess- Netzwerkssynthese (PNS), die ursprünglich zur konzeptuellen Entwicklung chemischer Prozesse entwickelt wurde, ein geeignetes Werkzeug zur Generierung und Analyse struktureller Alternativen für manufactural, Logistik und Geschäftsprozesse darstellt. Erweiterungen des ursprünglichen methodischen Rahmens zur Berücksichtigung spezifischer Randbedingungen von manufactural, Logistik und Geschäftsprozessen können jedoch die praktische Anwendbarkeit dieser Methode verbessern.

Das Ziel meiner Forschungsarbeit ist die Entwicklung einer Methode zur adäquaten mathematischen Abbildung und Modellierung von Lieferketten und der Synthese von optimalen und alternativen Liefer-Szenarios, die gegenseitigen Ausschluss, Zeitlimitationen und strukturelle Redundanzen berücksichtigen. Die vorgeschlagene Methode basiert auf den kombinatorischen Fundamenten der algorithmischen Prozesssynthese im Sinne der P-Graph Methode.

Viele praktische Anwendungsfälle können in Form von Beschränkungen und gegenseitiger Unvereinbarkeit von Aktivitätssätzen formuliert werden. Diese Beschränkungen und Rahmenbedingungen helfen, den Suchraum einzuschränken und damit die Suche nach strukturellen Alternativen wesentlich zu beschleunigen. Daher kann ein spezifischerer Satz von Alternativen in kürzerer Rechenzeit generiert werden.

Ein innovativer Algorithmus generiert das mathematische Modell und alternativen Szenarios mit erhöhter Zuverlässigkeit. Die Hauptschritte der Berechnung sind die Erstellung der Strukturen und die Bewertung der Zuverlässigkeit der einzelnen Liefer-Szenarien.

Die Erfüllung von zeitlichen Limitationen ist ein Schlüsselelement von manufactural, Logistik und Geschäftsprozessen und Lieferketten. In der vorliegenden Arbeit werden Zeitbeschränkungen in der Verfügbarkeit von Ressourcen, für Aktivitäten und Zeitzielsetzungen in manufactural, Logistik und Geschäftsprozessen berücksichtigt und in das mathematische Modell integriert und werden daher auch in den Ergebnissen der PNS Berechnung widergespiegelt.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Bertók Botond docens úrnak, folyamatos útmutatásáért és támogatásáért, mellyel a bemutatásra kerülő eredményeim és PhD dolgozatom megszületését segítette. Köszönetet mondok Dr. Friedler Ferenc professzor úrnak a szakmai együttműködésért. Köszönöm minden kollégámnak a kreatív, együttműködő és jó hangulatú légkört amiben dolgozhattam. Mindezek felett szeretném megköszönni szüleimnek azt a céltudatos, elszánt és kitartó ösztönzést és támogatást, mellyel tanulmányaim során elkísértek.

1. fejezet

Bevezetés

A dolgozat témája a gyártási, logisztikai és üzleti folyamatok strukturális alternatíváinak algoritmikus szintézise. Ezen folyamatok összetettek, több lépésből állnak, és a lépések összetett rendszere vezet egy folyamat előfeltételeitől annak eredményéig. A szintézis célja a feladatban definiált építőelemekből kívánt tulajdonságú folyamatok összeállítása. A szintézis a folyamat lépéseinek kiválasztását és azok kapcsolatainak meghatározását jelenti.

Egy gyártási, logisztikai vagy üzleti folyamathálózat és annak minden lépése jellemezhető a szükséges bemeneteivel (előfeltételeivel) és a lehetséges kimeneteivel (eredményeivel). Folyamathálózat-szintézis esetén feltételezzük, hogy a lépések kapcsolata kizárólag be- és kimeneteiken keresztül valósul meg. Két lépés akkor kapcsolódhat egymáshoz, ha egy lépés valamely kimenete (eredménye) biztosíthatja a másik lépés valamely szükséges bemenetét (előfeltételét). Az a kérdés, hogy egy lépés kimenete kielégítheti-e egy másik lépés szükséges bemenetét, bizonyos feladatok esetén könnyen megválaszolható, más feladatok esetén kevésbé. Ennek megfelelően a szintézis feladatok algoritmikus megoldása különböző nehézségekbe ütközik.

1.1. Célkitűzések

A dolgozatban bemutatom hogyan alkalmazható a P-gráf módszertan a gyártási, logisztikai és üzleti folyamatok strukturális alternatíváinak algoritmikus szintézisére.

Ahhoz, hogy minél szélesebb körben adaptálható legyen gyakorlati feladatok megoldására, a bővítni kell a folyamatszintézis során vizsgált feltételek körét ugyanúgy, mint az algoritmusokat melyek a feltételek vizsgálatára alkalmasak szintézis során:

- Egyik vizsgálandó feltétel logikai típusú, tehát strukturálisan is vizsgálható. Arra vonatkozik, hogy bizonyos alternatív tevékenységek külön-külön igen, de együtt nem szerepelhetnek egy megengedett megoldásban sem, tehát kölcsönösen kizárják egymást. Ennek oka lehet egy folyamat irányíthatósága, helyigénye vagy technológiai összekapcsolhatósága. Ilyen igény gyakran megjelenik üzleti folyamat tervezésben is, például üzleti folyamatok leírására egyre többször használt üzleti folyamat diagramokban (BPD) külön grafikus elem tartozik a kölcsönös kizáráshoz.

- Másik vizsgálandó szempont a folyamatok megbízhatósága, elsősorban strukturális szempontból. Tehát nem arra keresünk választ hogy egy folyamat egyes elemi tevékenységeinek a megbízhatósága hogyan befolyásolható, hanem, hogy ismert megbízhatóságú elemi lépésekből, hogyan építhetőek minél megbízhatóbb hálózatok.

- A harmadik szempont, mely szerint a PNS algoritmusok bővítését célul tűztük ki pedig az időbeli korlátok. A feladat olyan folyamathálózatok szintézise, melyek az erőforrások időben korlátozott rendelkezésre állása mellett is határidőre végetérnek.

Célom nem csupán egy-egy konkrét feladat megoldása, hanem olyan módszertan bemutatása, amely bármely az előbb említett feladatosztályokba (kölcsönös kizárást tartalmazó, redundáns és időkorlátos hálózatok) tartozó szintézis feladat esetén azok algoritmikus megoldását lehetővé teszi. Megmutatom – a gyakorlatban fontos szintézis feladatok megoldása során –, hogy melyek a különböző osztályba tartozó szintézis feladatok ismérvei, és milyen eszközöket használhatunk azok algoritmikus megoldására.

1.2. Saját eredményeim kiemelése

A dolgozat tartalmi részében – hagyományainknak megfelelően – mindvégig többes szám első személyt használok. Annak érdekében, hogy a dolgozatban elkülönítsem mások szakirodalomból ismert eredményeitől sajátjaimat, másokéra a szerzők nevével hivatkozom, sajátjaimat pedig minden fejezet és a dolgozat összefoglalásában egyes szám első személyben egyértelműen megfogalmazom.

1.3. Jelöléstan

A dolgozat számos formális leírást tartalmaz. Ezen leírások könnyebb olvashatósága érdekében egységes jelölésmódot használok a következők szerint:

- Egész vagy valós számot tartalmazó változót rendre dőlt kisbetű azonosít. Például x, y, z, i, j, v_1, v_i. Kivétel az f és a g, amelyek függvények.
- Halmazt kalligrafikus dőlt kisbetű, nagybetű vagy ezek sorozata jelöl. Például $\mathcal{m}$, $\mathcal{o}$, $\mathcal{x}$, $\mathcal{y}$, $\mathcal{A}$, $\mathcal{V}$, $\mathcal{M}$, $\mathcal{P}$, $\mathcal{inc}$, $\mathcal{exc}$. Halmazt reprezentál továbbá α és β a P-gráfok hagyományos leírásában.
- Relációt görög dőlt kisbetű azonosít. Például φ, ψ, ν, ω, φ^+, ψ^-. Kivétel az α és a β, amelyek halmazok a P-gráfok hagyományos leírásában.
- Vektort kövér kis- vagy nagybetű jelöl. Például: $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{e}_i$, $\mathbf{E}$.
- Függvényt pedig f vagy g dőlt kisbetű jelöl. Például: f, g, f^+, g^-.

2. fejezet

Irodalmi áttekintés

A most következő fejezetben néhány, az irodalomban megtalálható strukturális leíró módszer kerül bemutatásra, majd pedig a kutatásom alapjául szolgáló P-gráf módszertant ismertetem.

2.1. Folyamatok logikai struktúráját leíró módszerek

A következő alfejezetekben megismerkedhetünk az üzleti folyamatok modellezésével, a Petri-hálókkal, az állapot-taszk hálózatokkal, az erőforrás-taszk hálózatokkal, és végül az S-gráf módszertannal, amelyek a folyamatok különböző szempontok szerinti logikai struktúráját leíró módszerek.

2.1.1. Üzleti folyamat modellezés

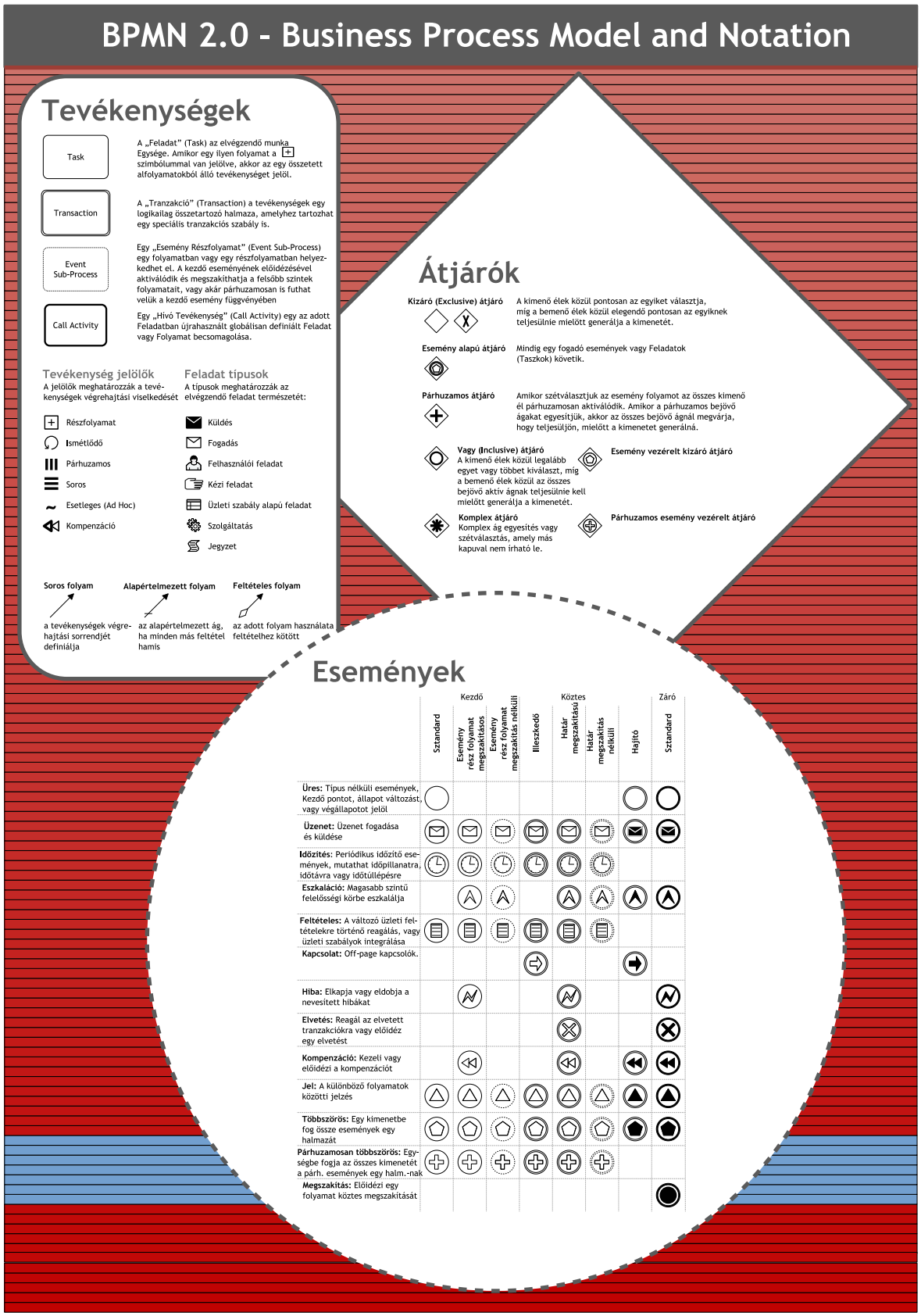
Ahhoz, hogy a különböző vállalatok és szervezetek a mai rohanó modern világban megőrizzék illetve növeljék üzleti sikereiket elengedhetetlen a üzleti folyamataik folyamatos ellenőrzése, és ha szükséges módosítása. A vállalatok számára kulcsfontosságú az üzleti folyamataik jól megalapozott tervezése és menedzselése, hogy hatékony versenytárs válhasson belőlük napjaink gyorsuló üzleti környezetében. Az üzleti folyamataik optimalizálására és folyamatos fejlesztésére fókuszálva az egyes szervezetek képesek lehetnek csökkenteni a működési költségeiket, növelni a minőséget és a hatékonyságot, és alkalmazkodni a folyamatosan változó körülményekhez, amely előnyhöz

juttatja őket az üzleti versenyben.

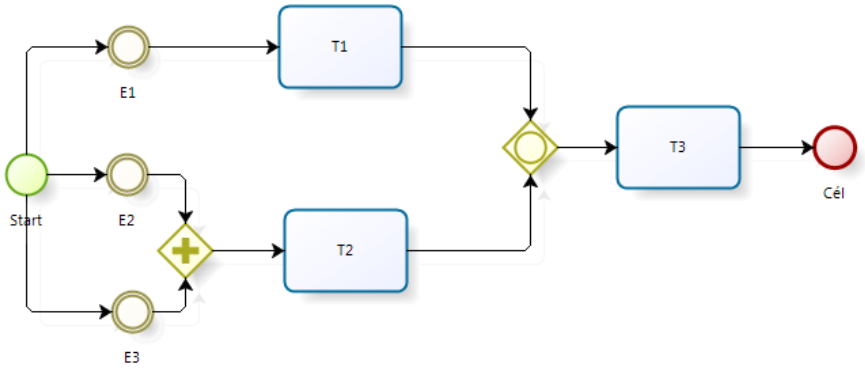
Az üzleti folyamatirányítás néhány évtizedes múltra tekint vissza. Napjainkban egyre több vállalat fedezi fel, hogy milyen sokat profitálhat abból, ha kellő részletességgel ismeri és érti a belső és külső folyamatait. 2001-ben egy 58 fős kutatócsoport létrehozta a Business Process Management Initiative-ot (BPMI), ami 35 különböző céget, szervezetet illetve magánszemélyt képviselve kezdte el működését. Jelenleg olyan jelentős cégek vesznek részt benne, mint például az IBM, az Oracle vagy az SAP AG. Munkájuk eredményeként 2002-ben publikálták a BPMN (Business Process Modell and Notation) 0.9 draft első kezdetleges verzióját, amelynek alapja az UML tevékenység diagramjából, az IDEF-ből és más, akkor már elfogadott jelölésrendszerekből ered. 2003 augusztusában jelent meg a BPMN 1.0 draft első nyilvános változata, majd 2004 májusában látott napvilágot a BPMN 1.0 verziója. A BPMI egyesült az OMG-vel (Object Management Group) 2005-ben, majd 2008-ban az OMG kiadta a BPMN 1.1 verzióját. Ezek után 2009 januárjában fogadták el a BPMN 1.2-est, majd 2009 augusztusában már kiadták a BPMN 2.0 béta 1 verzióját, melyet 2010 júniusára finomítottak, és megjelent a béta 2 verzió. A 2.0 verzió 2011 januárjában készült el, amely a jelenlegi legfrissebb kiadás ([5], [42], [77]). A 2.1 ábrán láthatóak a BPMN 2.0 legfőbb építőelemei ([16]).

A BPMN egy olyan szabványos grafikus leíró formátum, amivel a szervezetek szabványos módon egy üzleti folyamatmodellben írhatják le üzleti folyamataikat (Business Process Modell, BPM) ([22], [29], [49], [78], [97], [110]). A BPMN a BPD-t (Business Process Diagram) mint grafikus szemléltető eszközt specifikálja, amelyet a 2.2 ábra szemléltet. Az egységes és leegyszerűsített leírásnak köszönhetően az üzleti folyamatot leíró diagram jól definiálhatóvá válik, és megértése is egyszerűbb lesz, mind a külső, mind a belső feladatokra. Alkalmazásával e folyamatok szabványos módon írhatók le, és a grafikus megjelenítéssel jól értelmezhetővé válnak nemcsak más, BPD-t használó szervezetek, hanem a témában kevésbé jártas emberek számára is.

Számos, a BPMN szabványt támogató program létezik a kereskedelemben, amelyek szinte mindegyike képes az üzleti folyamatok grafikus ábrázolására és dokumentálására. Emellett elemző kérdésekre is választ kaphatunk a segítségükkel, bár a



2.1. ábra. Az üzleti folyamat modellezés főbb építő elemei



2.2. ábra. Az üzleti folyamat diagram, azaz BPD bemutatására szolgáló példa

kezelhető kérdések nehézségének a szintje széles sávban változhat az egyes implementációkban. Az üzleti folyamatok utólagos manuális optimalizálása ezeknek az elemző kérdéseknek a megfogalmazásával lehetséges, amelyek gyakorta nem egyértelműek és a rájuk kapott válaszok sem mindig azok, illetve nagyban függnek az alkalmazást kezelő gondolkodás menetétől és tapasztalataitól.

Ahogy később látni fogjuk az imént bemutatott BPMN és a későbbiekben ismertetésre kerülő PNS módszertanok feladatosztályai között jelentős különbségek vannak. Míg a BPMN-el főleg folyamatanalízis és -ábrázolás a cél, addig a PNS matematikailag bizonyítottan valamilyen szempont (például: költség, idő, megbízhatóság, stb.) szerint optimális folyamathálózatokat képes generálni.

2.1.2. Petri háló

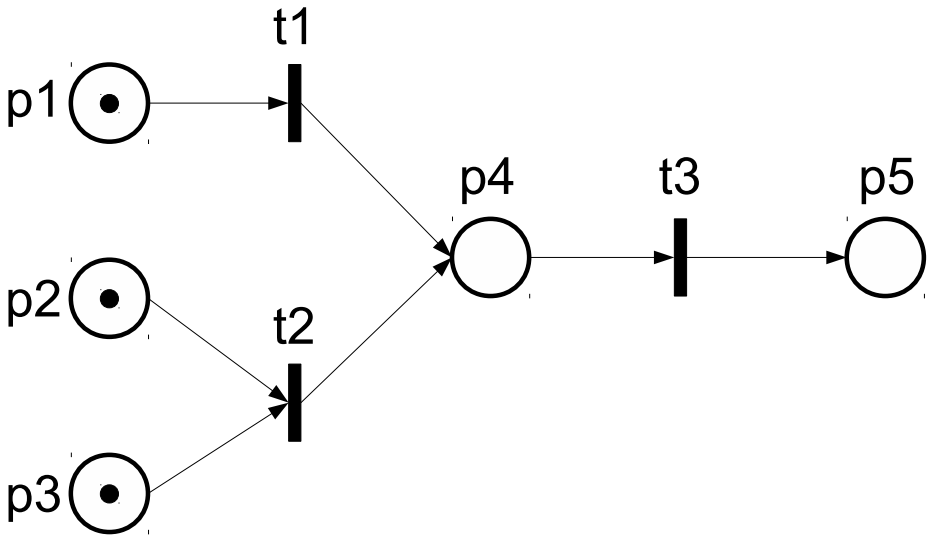
A Petri háló a rendszermodellezés és -analízis széles körében használható leíró eszköz ([19], [82], [84], [32]). Carl Adam Petri német matematikus 1939-ben dolgozta ki a Petri hálót (PN, Petri Net), amelyet eredetileg kémiai folyamatok leírására szánt. A matematikai alapokat 1962-ben a doktori disszertációjában dolgozta ki ([83]). A Petri hálók alkalmazási köre nagyon széles. Használható konkurens, aszinkron, elosztott, párhuzamos, nem-determinisztikus, sztochasztikus rendszerek modellezésére. A Petri háló kompakt módon fejezi ki a különböző állapotokat és szemléletesen fejezi ki a

szinkronizációt. Tömörebb, átláthatóbb modellek készíthetők általa.
A Petri hálók alapvető tulajdonságai a következők:

- Egyidejűleg grafikus és matematikai reprezentáció, mely által áttekinthető, precíz és egyértelmű.
- Strukturális leírás.
- Más ábrázolásmódok is kiteríthetőek Petri hálóvá.
- Egyszerű feladathoz is nagy Petri háló tartozhat.

A Petri háló egy absztrakt leírás, amely egy nem-determinisztikus véges automataként értelmezhető, strukturálisan pedig egy irányított, súlyozott, páros gráf ([19], [82], [84], [32]). A struktúrája követi a feladat logikai dekompozícióját és az egy-egy hely egy-egy logikai feltétel elvét valósítja meg. A strukturális leírás egy ötössel adható meg $PN = (\mathcal{P}, \mathcal{T}, \mathcal{E}, \mathcal{W}, M_0)$, ahol $\mathcal{P}$ a helyek (feltételek) halmaza, $\mathcal{T}$ az átmenetek (események, tüzelések, tranzíciók) halmaza, és $\mathcal{P} \cap \mathcal{T} = \emptyset$. Az $\mathcal{E} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P}) = \mathcal{I} \cup \mathcal{O}$ az élek halmaza, ahol az $\mathcal{I} : \mathcal{P} \to \mathcal{T}^{\infty}$ a bemeneti (előfeltétel) függvények halmaza és az $\mathcal{O} : \mathcal{T} \to \mathcal{P}^{\infty}$ pedig a kimeneti (következmény) függvények halmaza. A $\mathcal{W} = \cup w_i, i = 1, \ldots, |\mathcal{E}|, w_i : \mathcal{E} \to \aleph^+$ az élek súlyfüggvényeinek a halmaza. A $w_i(e), i = 1, \ldots, |\mathcal{E}|$ súlyú e él ugyanazt jelenti, mintha w_e darab párhuzamos élt használnánk. Az egyszeres súlyokat nem szokás feltüntetni grafikusan. Az $\mathcal{M}_0 : \mathcal{P} \to \aleph$ pedig a kezdőállapotok halmaza.

Grafikusan egy irányított páros gráffal ábrázolható, ahol a csúcsok a helyek ($\mathcal{P}$) és az átmentek ($\mathcal{T}$), az élek pedig a bemeneti ($\mathcal{I}$) és kimeneti ($\mathcal{O}$) függvények. Az egyes helyek állapotait a benne levő tokenek száma határozza meg. A tokeneket más néven állapotjelölőket vagy jelölőpontokat fekete pöttyel jelöljük az adott hely típusú kör alakú csomópontban. A 2.3 ábra egy petri hálóval ábrázolt folyamatot szemléltet. Egy hely típusú csomópontban a leírt rendszer aktuális állapotának megfelelően tetszőleges számú token lehet. Az egyes helyek token számát a $\mu : \mathcal{P} \to \aleph$ jelölőfüggvény adja meg, ahol $\mu_i = \mu(p_i) \geq 0$ komponense a p_i helyen található tokenek száma. A hálózat állapotát az egyes helyek állapotainak összességével, azaz a $\mu^T = [\mu_1, \mu_2, \ldots, \mu_n]$ ($n = |P|$) állapotvektorral jellemezzük.



2.3. ábra. A petri hálót szemléltető példa

A petri háló működése során állapotváltások történnek, amikor is az egyes átmenetek „tüzelnek” vagyis működnek. Egy állapotváltás akkor következhet be, ha az előfeltételek igazak, azaz van elég token a bementi helyeken. Minden egyszeres él egy tokent szállít az adott helyre. Adott i-edik átmenet „tüzelését” a következőképpen írhatjuk fel: $\underline{\mu}^{(i)}[t_j > \underline{\mu}^{(i+1)}$. Tüzelés után a következmények lesznek „igaz”-ak. Azaz átmenetek bemeneti tokenjei újraosztásra kerülnek a kimeneten. Ezek alapján a „tüzelési” vagyis működési sorozat a következőképpen adható meg: $\underline{\mu}^{(0)}[t_{j0} > \underline{\mu}^{(1)}[t_{j1} > \ldots [t_{jk} > \underline{\mu}^{(k+1)}$.

Petri háló kiterjesztések

A Petri hálók szintaxisát és szemantikáját az idők folyamán számos irányba bővítették. Ezek közül jónéhány maguknak a Petri hálóknak egy-egy speciális problémakörre történő szűkítésén, valamint a problémakör sajátosságainak megfelelő, de más alapvető szintaktikai és szemantikai elemekből levezethető jelölésbeli gazdagításán alapulnak. Az alap Petri hálóban a helyek és az átmenetek diszkrétek, azaz a helyek csak egész számú tokeneket tartalmazhatnak míg az átmenetek csak egész számszor tüzelhetnek. A következőekben a teljesség igénye nélkül bemutatásra kerül néhány

Petri háló kiterjesztés.

Folytonos vagy Folyékony Petri háló (Continuous/Fluid PN)

A folytonos Petri hálókban a helyek valós értékeket vesznek fel, és az engedélyezett átmenetek tüzelhetnek bármely valós értékre, ha rendelkezésre áll a megfelelő számú bementi token, ahogy az David, Silva és szerzőtársaik munkásságaiban található ([3], [23], [25], [40], [89], [90], [99], [100], [101], [102], [107]). A helyeket és az átmeneteket ebben az esetben folytonosnak nevezzük. A hálózat súlyfüggvényei nem negatív egészek, ugyanúgy mint az alap Petri hálók esetén. Egy folytonos Peti háló értelmezhető úgy mint egy diszkrét Petri háló folyékony relaxációja. A továbbiakban egy ν átmenet bemeneteit $\bullet\nu$-vel, a kimeneteit pedig $\nu\bullet$-vel jelöljük.

Definíció 2.1.1 *Egy folytonos Petri háló egy $< N, m_0 >$ párral írható le, ahol N a Petri háló, az $m_0 \in \Re_{\geq 0}^{|P|}$ pedig a kezdő állapot. Az evolúciós szabály abban különbözik a diszkrét Petri hálókhoz képest, hogy a folytonos Petri hálókban a tüzelés nincs egész értékekhez kötve, és az egyes állapotoknak $m \in \Re_{\geq 0}^{|P|}$ nem kell egésznek lenniük. Egy átmenet t_i akkor és csak akkor engedélyezett az m állapotban, ha minden $p_j \in \bullet t_i$, $m[p_i] > 0$; és az engedélyezési foka $enab(t_i, m) = min_{p_j \in \bullet t_i}\{m[p_j]/Pre[p_j, t_i]\}$. A t_i tüzelése egy bizonyos $0 \leq \alpha \leq enab(t_i, m)$ mennyiségre egy új $m' = m + \alpha * C[P, t_i]$ jelöléshez vezet, ahol $C = Post - Pre$ a token folyam vagy incidencia mátrix, és $C[P, t_i]$ jelöli a C t_i-nek megfelelő oszlopát.*

Hibrid Petri háló (Hybrid PN, HPN)

Alla és David bevezették a hibrid Petri háló fogalmát, amely tartalmazhat folytonos és diszkrét helyeket ([3], [4], [24], [25]). Továbbá míg a folytonos Petri hálókban minden átmenet folytonos addig a hibrid Petri hálókban lehetnek szigorúan diszkrét és folytonos átmenetek is. DiFebbraro és szerzőtársai egy hibrid Petri háló alapú városi forgalom irányító rendszert dolgoztak ki, ahol a gépjármű forgalmat egy idővezérelt modellel, míg a forgalomirányító lámpák vezérlését egy diszkrét eseményű modellel írják le ([27]). Julvez és szerzőtársai hatékony Matlab toolbox-ot dolgoztak ki hibrid petri hálók tervezéséhez, analíziséhez és szimulációjához ([61]).

Időzített Petri háló (Timed PN, TPN)

Annak érdekében, hogy modellezni és analizálni lehessen a valós dinamikus rendszerek teljesítményét és ütemezhetővé váljanak, az idő fogalmát is bevezették a Petri hálók formalizmusába, ahogy az Berard, Boyer és Bucci munkásságaiban is látható ([106], [10], [11], [15], [17], [81], [109]). Ezekben a feladatokban az idő késleltetésként kerül bevezetésre a helyeken, az átmeneteknél és a tokeneknél is. Számos variációt dolgoztak ki a időzített Petri hálókra. Ezek a variációk abban is különbözhetnek egymástól, hogy az átmenetek tüzelését hogyan értelmezik. A fő kérdés az, hogy a bemeneti tokenek azonnal vagy csak az átmenetek tüzelési idejének lejárta után törlődnek.

Determinisztikus időzített Petri háló (Deterministic Timed Transitions PN, DTTPN)

Ramchandani a determinisztikus időzített Petri hálók esetén az összes átmenethez determinisztikus idő címkéket társított, amelyek a tüzelések időhosszait jelölik ([88]). A hálózat minden más tulajdonsága azonos az alap Petri hálókéval. Amennyiben egy átmenet engedélyezett a DTTPN-ben és ki is lett választva a tüzelésre, akkor a tokeneket eltávolítják a kezdeti időpillanatban a bemenetről és az átmenet megkezdi a működését a hozzárendelt időtartamig. Miután az átmenet működési ideje letelt a megkövetelt tokenek a kimenetre kerülnek. Így az egymással rivalizáló átmenetek nem tudják akadályozni egymást azáltal, hogy ellopják a bementi tokeneket a tüzelés alatt.

Általánosított sztochasztikus Petri háló (Generalized Stochastic PN, GSPN)

Marsan és szerzőtársai szerint a sztochasztikus Petri háló egy olyan háló, ahol az összes átmenet időzítését egy exponenciális eloszlású változó definiálja ([20], [72], [73], [74]). A valószínűségi változóval definiált késleltetés az átmenet engedélyezettsége és tüzelése közötti idő. Egy korlátozott sztochasztikus Petri háló elérhetőségi gráfja izomorf egy véges Markov lánccal, amely lehetővé teszi az eredeti rendszer teljesítmény analízisét. Egy általánosított Petri háló tartalmazhat időzített és azonnali átmeneteket. A GSPN-ekben a tokenek csak az átmenet késleltetésének letelte után törlődnek a

bemeneti helyekről, ennek következményeként például előfordulhat, hogy az átmenet tüzelése lehetetlenné válik egy másik átmenet tüzelése miatt.

Árazott és időzített Petri háló (Priced Timed PN)

A valós rendszerek modellezésében a műveletek idő és a költség paraméterei egyaránt fontosak. Abdulla és Mayr a költséget az időzített Petri hálókba egy a helyekből és átmenetekből a természetes számok halmazára leképezett költség függvényként vezette be ([1]). Egy hely költsége az adott helyen egy token tárolásának a költsége adott időegységenként, egy átmenet költsége pedig az adott átmenet végrehajtásának a költsége. Az elmúlt két évtizedben az elérhetőségi probléma Petri hálóval történő megoldásának irodalma intenzíven növekedett, annak ellenére, hogy nincs hatékony Petri háló alapú algoritmus, ami meg tudná oldani az ebbe a témakörbe tartozó problémákat néhány speciális feladatosztályt leszámítva. Igaz számos, a gyakorlatban jól alkalmazható algoritmus létezik, de ezek mindegyike vagy költség vagy idő szerint keresi az optimális megoldást.

Színezett Petri-háló (Colured PN, CPN)

A színezett Petri-hálót Jensen dolgozta ki ([60]) 1981-ben. A tokenek által reprezentált különböző fogalmak egymástól való megkülönböztethetőségét teszi lehetővé a különböző színek használata ([106], [70]). Így a tokenek típusok szerint csoportosíthatóak, továbbá különböző tulajdonságok rendelhetők hozzájuk. Ennek következtében az átmenetekhez járulékos szabályok, többek között előfeltételek rendelhetőek, amelyek a az átmenetek által reprezentált tevékenységek végrehajtását határozzák meg. A tokenek szinkronizálására szintén használhatunk előfeltételeket, amelyek segítségével meghatározható, hogy az átmenet akkor és csak akkor tüzelhet, ha a megfelelő színű (típusú) tokenből, megfelelő számú áll rendelkezésre. A modell sokkal egzaktabbá és szélesebb körben alkalmazhatóvá válik a Petri-háló ezen irányú kiterjesztésével. Viszont ebben az esetben a grafikus reprezentáció már nem tartalmazza azon információk összességét, amelyek a Petri háló alap leírásában megtalálhatóak. Az egyes tevékenységeket leképező átmenetek leírásához pedig egy pszeudókódos leírásnak is

tartoznia kell a színezett kiterjesztés esetén.

Elérhetőségi probléma

A Petri háló analízis egyik kiemelt problémája az úgynevezett elérhetőségi probléma, amely azt vizsgálja, hogy egy bizonyos állapot elérhető-e a kiinduló állapotból ([53], [32]). Ilyen tipikus problémák a vezérlési problémák, illetve a biztonsági problémák, amikor egy nem kívánt állapot elérhetőségét vizsgáljuk egy biztonságos állapotból kiindulva. A vegyipari termelési folyamatok modellezésére és optimalizálására kidolgozott folyamat-hálózat szintézis (Process Network Synthesis, PNS) egy matematikailag erősen rokon problémakör. A PNS feladat során adott nyersanyagok felhasználásával adott mennyiségű végtermék gyártása a cél bizonyos műveleti egységek segítségével. A feladat nem csupán egy alkalmas hálózat megadása, hanem a valamilyen szempont szerint optimális hálózat megadása is. A PNS módszertan a 2.2 fejezetben részletesen bemutatásra kerül ezért sem ebben sem a következő alfejezetben erre nem kerül sor.

A PNS és a Petri háló összehasonlítás

Mivel a 2.1 fejezetben bemutatásra kerülő strukturális leíró módszerek közül egyedül a Petri háló elérhetőségi problémája mutat hasonlóságot a 2.2 fejezetben bemutatásra kerülő PNS módszertannal, amely kutatásom alapját képezi, ezért ezen két módszertan összehasonlítását mutatom be részletesebben. A Petri hálónak és a PNS módszertanban használt P-gráfnak hasonló a megjelenése, azaz mindkettő egy irányított páros gráfon alapszik. Továbbá az átmeneteknek ugyanaz a szerepük, mint a műveleti egységeknek vagyis a bemeneteiken megjelenő entitásokat átalakítják a kimeneti entitásaikra. A Petri háló optimális elérhetőségi feladatának a célja és a PNS probléma erős hasonlóságot mutatnak egymással. Ez pedig a következőképpen írható le. A kezdő állapotból kiindulva adott egy optimális útvonal, ahhoz hogy elérjük a meghatározott végállapotot. A 2.1. táblázat mutatja be a Petri háló elérhetőségi problémája és a PNS módszertan közötti leképzést, amely leképezéssel már Gyapay és Pataricza is foglalkozott korábban ([53]).

2.1. táblázat. A PNS feladat és a Petri háló elérhetőségi feladata közötti leképezés

Optimális elérhetőségi feladat Petri háló felhasználásával	**Folyamat-hálózat szintézis (PNS) feladat**
helyek (p_i)	anyag jelegű csomópontok (m_i)
átmenetek (t_j)	műveleti egységek (o_j)
M_0 tokenek	nyersanyagok (R)
$M_{partial}$ helyek	végtermékek (P)
átmenetek költsége	műveleti egységek működési költsége
az élek súlya	a műveleti egységek felhasználási és előállítási arányai
egy adott hely tokenjeinek a száma	az egyes anyagok aktuális mennyisége
az optimális elérhetőség karakterisztika vektora	a PNS feladat megoldás struktúrája
a karakterisztika vektor feszített (spanned) Petri hálója	a megoldás struktúra P-gráfja
egy optimális elérhetőségi feladat Parik vektora	a PNS feladat MILP problémájának egész megoldása

Igaz a Petri háló elérhetőségi problémának és a PNS feladatnak a célja és a matematikai leírása is hasonló, mégis alapvető különbségek vannak az általuk képviselt modellezendő területek és az elvárt megoldás között. Ezen különbségek az alkalmazási területeik különbözőségéből fakadnak. Míg a Petri hálókat főleg a hálózatok analízisére, validációjára és szimulációjára használják, addig a PNS módszertan a valamilyen szempont (például: költség, idő, megbízhatóság, stb.) szerint optimális hálózat meghatározásra használható.

Különbség mutatkozik a módszertanok között a vagy kapcsolat kezelésében is. Míg a Petri hálok esetében a vagy kapcsolatok kezelése véletlenszerű, addig a PNS módszertanban mi hozunk arról döntést, hogy a vagy kapcsolat melyik ágán haladunk tovább. Összehasonlítva ezeket a korábban bemutatott BPMN jelölés rendszerrel, ott a vagy kapcsolat az nem a hagyományos értelemben vett alternatíva, azaz általában nem mi hozunk döntést arról, hogy merre haladunk tovább, hanem az valamely külső tényezőtől függ.

A P-gráf egy folytonos modell, míg a Petri háló alap értelmezése diszkrét. Igaz némely Petri háló folytonossá tehető bizonyos megszorítások mellett, de ezen megszorítások eredményeképpen rengeteg tulajdonsága elvész. A P-gráf megadja az adott feladat valamilyen szempont szerinti (költség, idő, megbízhatóság, stb.) optimális megoldását, viszont arról nem ad információt, hogy elindítható-e a megoldás. A Petri háló ezzel ellentétben meg tudja mondani, hogy egy megoldás elindítható-e, használva az elemző, validáló és szimuláló eszköztárát, viszont nem képes automatikusan optimális megoldást előállítani. Míg a P-gráf módszertan egy meghatározott optimális cél felé halad a megoldás során, addig a Petri háló attól függően, mekkora a rugalmasság a rendszerben elég távol is kerülhet az optimális megoldástól, továbbá a precedencia megadás lehetősége tovább torzíthatja a kapott megoldást az optimalitás szemszögéből. Ezeket figyelembe véve a két módszertant egymással kiegészítve egy nagyon hatékony eszköztárat kaphatunk, ahol is a P-gráf módszertannal meghatározott optimális megoldást a Petri háló eszköztárában található elemző eszközökkel validálnánk. Ez utóbbi felvetés a későbbiekben esetleg további érdekes eredményekre vezethet, de jelen dolgozatnak nem képezi a tárgyát.

2.1.3. Állapot-tevékenység hálózat (STN)

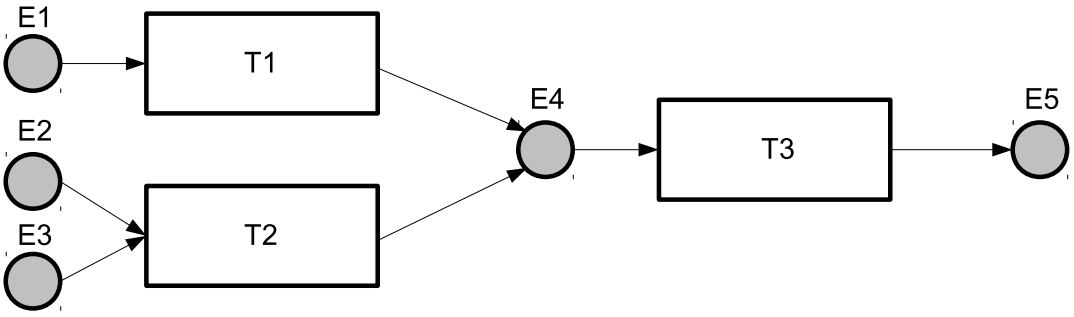
Az állapot-tevékenység hálózat (State Task Network, STN) leírást Gabriela Garcia-Ayala fejlesztette ki Ignacio E. Grossmann vezetésével többtermékes batch üzemek ütemezésére. Az STN-hez Kondili és társai kidolgoztak egy vegyes egész lineáris programozási (mixed integer linear programming, MILP) modellt szakaszos működésű kémiai folyamatok rövid távú ütemezéséhez ([66]). A 2.4 ábrán látható módon a nyersanyagokat, a köztes termékeket és a végtermékeket az állapotok (state), valamint a műveleteket – amelyekben az egyik állapot egy másikba alakul át – a tevékenységek (task) jelölik.

Jelölés	Leírás
T1	Taszk csomópont: az olyan végrehajtó műveleteket szemlélteti, amelyek egy vagy több bementi állapotból előállítanak egy vagy többi kimeneti állapotot
E1	Állapot csomópont: jelöli a betáplálást, a köztes és a végtermékeket

2.4. ábra. Az állapot-tevékenység hálózat, azaz az STN leírás elemei

A 2.5 ábrán láthatunk példát az STN leírásra. Ebben a formalizmusban egyforma méretű darabokra osztják fel a rendszer működéséhez rendelkezésre álló időt (time horizon), ahol a tevékenységek kezdési és befejezési idejeinek meg kell egyeznie ezen diszkrét időpontokkal. Ebben a modellben a fő előny a nagyon általános folyamatok, receptek kezelésének lehetősége, beleértve például a tárolási típusokat, az anyagok körforgását és az erőforrásokkal kapcsolatos feltételeket. A megközelítésben a legnagyobb problémát a létrehozott MILP modell nagy mérete illetve a diszkretizálás pontjainak a meghatározása jelenti. Ez a munka alapjául szolgált sok más kutatásnak, ahol a fő cél a számítási teljesítmény növelése volt, felhasználva a leírás előnyeit. Sahinidis és szerzőtársai szétbontották a modellt úgy, hogy annak nagyobb mérete

ellenére is növelték a megoldás hatékonyságát ([92]). Shah és szerzőtársai a célfüggvény lehető legjobb közelítése érdekében változtattak a hozzárendelési szabályokon ([96]). Elkamel egy heurisztika segítségével szétbontotta a feladatot ([31]). Yee és Shah további heurisztikák beépítésének hatását vizsgálta ([111]).



2.5. ábra. Az állapot-tevékenység hálózat, azaz az STN leírást szemléltető példa

Mockus és Reklaitis bemutatva az időintervallumok és az események koncepcióját az STN leíráson alapuló folytonos idejű modellt készített ([75]). Modelljükben az időintervallumok reprezentálják a két esemény között eltelt idő hosszát, az események pedig a tevékenységek kezdetét és végét. A feladatot egy vegyes egész bilineáris programozási (mixed integer bilinear programming, MIBLP) modellel írták le. A modellt linearizálták úgy, hogy egy lineáris feltételekkel korlátozott MIBLP feladatot kapjanak, amit egy módosított külső közelítési (outer approximation) algoritmussal oldottak meg. A globális optimumot nem tudták garantálni és gyenge futási eredményeket értek el egy szakirodalmi példa esetében is. Ierapetritou és Floudas az STN leírást és az eseménypontokat használó folytonos idejű modellen alapuló módszert készített ([58]). Egy eseménypont reprezentálja egy tevékenység elkezdését vagy befejezését a rendelkezésre álló időintervallum egy tetszőlegesen választott időpillanatában, ahol az eseménypontok optimális számát egy iteratív módszerrel határozzák meg. Az előző formalizmustól való eltérés abban rejlik, hogy ebben a megközelítésben az eseménypontok nemcsak egy időpontot azonosítanak, hanem két eseménypont közötti távolságot is ezzel definiálják. A modell fő erénye az eredeti MILP feladat méretének nagy mértékű csökkentése volt, amit úgy értek el, hogy a berendezésekhez és a tevékenységekhez rendelt eseményeket szétválasztották; a berendezéseket nem a

tevékenységekhez rendelték, hanem az eseménypontokhoz. A módszer legfőbb hátránya, hogy nem tudja az optimumot biztosítani.

Igaz, hogy a jelen alfejezetben bemutatott STN leírás és a következő alfejezetben ismertetésre kerülő RTN leírás ábrázolási módja is hasonlatos a 2.2 fejezetben bemutatásra kerülő PNS módszertanban használatos P-gráffal, de a módszertanok által kezelhető feladatosztályok között szignifikáns különbségek vannak. Alapvetően az STN és az RTN leírás is többtermékes batch üzemek ütemezésére használható, azaz egy adott folyamat struktúrán belüli a tevékenységek optimális sorrendjének a meghatározására, addig a PNS módszertan segítségével maga a valamilyen szempont (például: költség, idő, megbízhatóság, stb.) szerint optimális folyamathálózat struktúra határozható meg. Továbbá az STN és az RTN sem képesek kezelni azt, ha egy tevékenységet egyidőben két berendezés összedolgozva végez el, és ekkor a feladatmegosztástól függ a szükséges idő. Tehát nem csak az a különbség, hogy nem a folyamatot optimalizálják, hanem hogy a tevékenységek méretét sem változtatják, míg a PNS feladatok megoldására kidolgozott P-gráf módszertannal kezelhetőek ezek az esetek is.

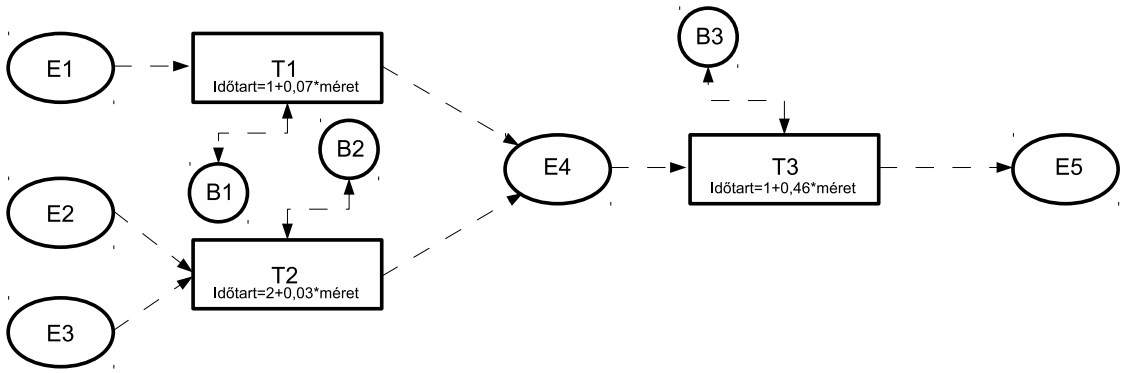
2.1.4. Erőforrás-tevékenység hálózat (RTN)

Az STN leírás és a hozzá tartozó formalizmus helyett Pantelides egy alternatív leírást ajánlott, az erőforrás-tevékenység hálózatot (Resource-Task Network, RTN), amely az erőforrások egységes leírásán alapult ([80]). Ebben a leírásban a tevékenységek nem anyagokat fogyasztanak és állítanak elő hanem erőforrásokat (resource). Erőforrásnak minősülnek a szállítási eszközök, az emberi erőforrások, a tárolás, az energia, a nyersanyagok, a köztes termékek és a végtermékek is. A tevékenységek definíciója megegyezik az STN leírásban használttal, kiegészítve a tárolással, a tisztítással és a szállítással. A 2.6 ábrán láthatóak az RTN leírásban használatos építőelemek a 2.7 ábrán pedig egy példát láthatunk az RTN leírásra.

Zhang és Sargent az RTN leíráson alapuló folytonos idejű modellt készített ([114], [113]). Ebben a formalizmusban a teljes időtartam változó, előre meg nem határozott méretű részekre van feldarabolva, ahol a tevékenységek kezdő és végidőpontjai adják

Jelölés	Leírás
T1	Taszk csomópont: az olyan végrehajtó műveleteket szemlélteti, amelyek egy vagy több bementi állapotból előállítanak egy vagy több kimeneti állapotot, kiegészítve a tárolással, a tisztítással és a szállítással
E1 vagy B1	Erőforrás csomópont: Erőforrásnak minősülnek a szállítási eszközök, az emberi erőforrások, a tárolás, az energia, a nyersanyagok, a köztes termékek és a végtermékek is

2.6. ábra. Az erőforrás-tevékenység hálózat, azaz az RTN leírás építő elemei



2.7. ábra. Az erőforrás-tevékenység hálózat, azaz az RTN leírást szemléltető példa

meg az időintervallumok szélességét. A matematikai formalizmus egy MINLP modellt eredményez, amelyet a Glover transzformációval linearizáltak és így egy MILP feladatot kaptak ([50]). Természetesen, mint minden linearizálási technika esetében, a feladat dimenziója jelentősen megnőtt és így a feladat hamar elér ahhoz a méretheterhatárhoz, amely felett már nehezen kezelhető az általánosan használt MILP megoldókkal. Hasonló módszert dolgozott ki Schilling és Pantelides azzal a különbséggel, hogy ők feltételezték, hogy egy tevékenység mérete (a végrehajtási ideje vagy az adott idő alatt előállított anyag mennyisége) nem függ a választott berendezéstől, valamint kikötötték, hogy a tevékenységek csak a működésük elején és a végén vannak közvetlen kapcsolatban az erőforrásokkal ([95]). A kapott MINLP feladatot szintén a Glover

transzformációval alakították át MILP feladattá, amit egy szétválasztás és korlátozás algoritmussal oldottak meg. Az algoritmus újdonsága abban rejlett, hogy nem csak az egész, hanem a folytonos változók szerint is végeztek szétválasztást.

Ahogy azt az előző alfejezetben már ismertettem, az STN és az RTN leírások, az ábrázolási módjaik hasonlósága ellenére, szignifikáns különbséget mutatnak a PNS módszertanhoz képest az általuk megoldható feladatok terén. Míg az STN és az RTN leírások többtermékes ütemezésére használhatóak, addig a PNS módszertan segítségével az optimális folyamathálózat struktúra határozható meg. Továbbá a tevékenységek méretét sem képesek változtatni abban az esetben, ha egy tevékenységet egyidőben két berendezés összedolgozva végez el, és ekkor a feladatmegosztástól függ a szükséges idő, míg a PNS feladatok megoldására kidolgozott P-gráf módszertan képes erre.

2.1.5. S-gráf

Termelő és szolgáltató rendszerek működését, viselkedését elemezve az esetek jelentős részében ütemezési feladatokkal találkozunk, így például a vegyiparban, az olajiparban, a gépiparban, a mezőgazdaságban, az építőiparban és a szállítmányozásban. A termelő és szolgáltató rendszerek mellett az informatikai rendszerekhez is kapcsolódnak bonyolult ütemezési feladatok. Ha a rendszer tartalmaz olyan konkurens folyamatokat, melyek ugyanazokat az erőforrásokat igénylik és nem áll rendelkezésre megfelelő számú erőforrás, akkor az erőforrások ütemezésével lehet a folyamatokat kiszolgálni.

Ütemezési feladatokban általában a következő elnevezéseket használják:

- **Termék** (product): az ütemezés során előállítandó anyag, tárgy, szolgáltatás.
- **Recept** (recipe): a termék előállítását leíró gyártási utasítások.
- **Batch** (batch): a termék előállításának egyszeri folyamata, mely során adott mennyiségű termék keletkezik. A shop ütemezési feladatoknál a munka (job)

elnevezést használják a szakirodalomban. A batch magyar nyelven köteget, adagot jelent, azonban a vegyiparban a francia eredetű sarzs (charge) szót használják.

- **Tevékenység** (task): a termék előállításának egy elemi lépése, mely során adott bemenetből adott kimenet keletkezik. A tevékenységeket a szakirodalomban szokás műveletnek, munkának is nevezni.

- **Berendezés** (equipment unit): a taszk végrehajtására használható eszköz. Szokás még gépnek is nevezni.

A vegyipari termékek jelentős részét szakaszos gyártási műveletekkel állítják elő. A szakaszos gyártási folyamatok legjelentősebb tulajdonsága a nagy mértékű rugalmasság, mellyel nagyszámú, különböző terméket lehet előállítani.

Termelő folyamatoknál fontos kérdés, hogy a gyártás során keletkező köztes anyagok milyen tulajdonságúak. Tárolás szempontjából kérdéses, hogy a köztes anyagokat lehet-e tárolni, kell-e tárolni, vagy a folyamat és a gyártási környezet olyan, hogy a köztes anyagok tárolási kérdéseivel nem kell foglalkozni az ütemezés során. Ha tárolni kell a köztes anyagokat, akkor azokat csak a tárolásra használt dedikált tároló berendezésekben, vagy magában az anyagot gyártó berendezésben lehet tárolni addig, amíg a következő gyártó berendezésbe nem kerül. Az ütemezési feladatokban többféle tárolási stratégia is előfordulhat. Ezek közül az UIS (Unlimited Intermediate Storage) tárolási stratégia kerül bemutatásra, amely esetén a gyártási folyamat alatt keletkező köztes anyagokat végtelen mennyiségben lehet tárolni. A szakirodalomban elsősorban az UIS stratégiával foglalkoznak, amely főként a gépiparra jellemző, ahol a köztes anyagok tárolása könnyen megoldható egy nagy raktárépület segítségével.

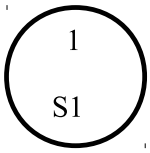
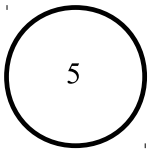
Az ütemezési feladatok két leggyakoribb célja a teljes működési idő minimalizálása, illetve a profit maximalizálása. A gyakorlatban azonban más nézőpontok is figyelembe vehetőek, mint megkötések vagy optimalizálási célok, mint például a szennyvíz minimalizálás, amelyre Gouws és Majozi mutat példát ([52]) vagy a hő visszanyerés maximalizálása, amely Adonyi és szerzőtársai munkásságában kerül bemutatásra ([2]).

Az ütemezési feladatokról már korábban bebizonyosodott, hogy algoritmuselméleti szempontból az NP nehéz feladatok közé tartoznak. Ezért különösen fontos egy célszerű modellezési technika és megoldó módszer kidolgozása. Lenstra és Rinnoy Kan bizonyította a hagyományos job shop feladatról, Tzafestas pedig a rugalmas termelő rendszerek (Flexible Manufactoring System, FMS) ütemezéséről, hogy azok az NP teljes feladatok osztályába tartoznak. Ha az open shop feladat két gépes, akkor létezik polinomiális algoritmus az optimális megoldás megkeresésére, ha a gépek száma több, mint kettő, akkor a feladat NP teljes. Ütemezési feladatok jelentős részéről bizonyítható, hogy NP teljesek, azaz más nehéz feladatokkal ekvivalensek. Az NP teljességet az ütemezési feladat egy másik NP teljes feladattá transzformálásával igazolták. Általában az ütemezési feladatok felírhatóak egy vegyes egész lineáris/nem lineáris programozási (MILP/MINLP) feladatként ([38]). Másrészről más módszerek speciális gráfokat használnak a matematikai modellhez, mint például az S-gráf módszertan ([94]) vagy az időzített automaták ([79]). Az ütemezési feladatot MILP/MINLP matematikai programozási modellként, vagy gráf alapú modellként felírva majd megoldva, a megoldás megkeresése egy NP nehéz feladat.

Az S-gráf módszertan (S-graph framework) szakaszos folyamatok optimális ütemezésének meghatározására bevezetett gráf ábrázolási mód – amelynek építő elemei a 2.8 ábrán láthatóak – és hatékony algoritmus ([93], [91], [94]). A módszertant az általános ütemezési feladatok megoldására hozták létre, azonban a szerzők lehetőséget biztosítottak speciális ütemezési feladatok S-gráf módszertannal történő megoldására, az alap módszertan feladatfüggő gyorsítási lehetőségeire. Az S-gráf módszertan hatékonysága az ütemezési feladatok kombinatorikus tulajdonságainak kihasználásán alapszik ([93], [94]).

A szakirodalomban sokfajta ütemezési feladattal találkozunk, melyek jellegükben, bonyolultságukban, a megoldásukra kidolgozott módszerekben jelentősen különböznek egymástól. Az S-gráf módszertanban az ütemezési feladatoknak azt az osztályát tekintjük, melyben a cél egy olyan optimális ütemezés megtalálása, mely képes a lehető legrövidebb idő alatt a kívánt mennyiségű termék előállítására a rendelkezésre álló szakaszos üzemű berendezések felhasználásával. A kívánt termékmennyiség előállításának teljes idejét végrehajtási időnek nevezzük. Minden termék tevékenységek

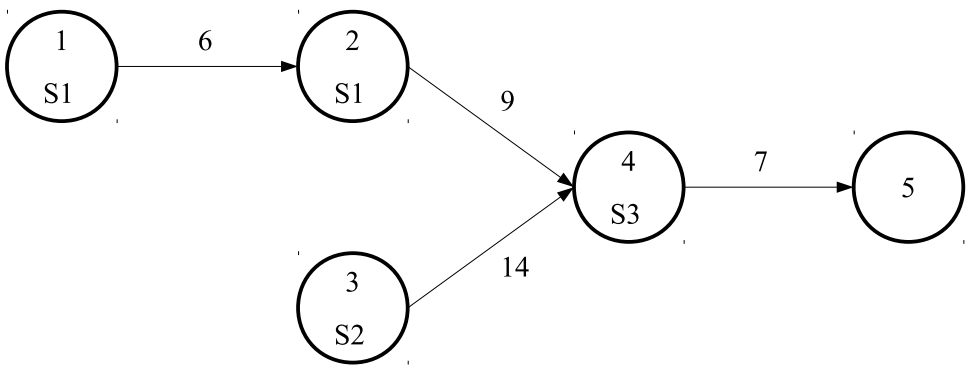
rögzített sorrendű hálózatával állítható elő. A gyakorlatban egy tevékenység általában több alternatív berendezéssel hajtható végre, amely berendezések nem feltétlenül ugyanannyi idő alatt végzik el az adott tevékenységet. Az ütemezési algoritmus feladata ezen berendezések közül minden tevékenységhez egy megfelelőt hozzárendelni, és a berendezésekhez rendelt tevékenységek végrehajtási sorrendjét, azaz a berendezések ütemezését meghatározni. Egy ütemezési feladat megoldását, azaz a berendezésekhez tartozó tevékenységek ütemezését egyértelműen megadhatjuk a berendezésekhez rendelt tevékenységeken megadott rendezés formájában. Az ütemezési algoritmus célja a berendezések olyan ütemezésének a meghatározása, mely a feladat optimális ütemezését adja.

Jelölés	Leírás
	Taszk-csúcs: a taszk-csúcsok tartalmaznak egy egyértelmű azonosítót (pozitív egész szám), illetve a hozzájuk rendelhető berendezések Si ($Si = \{Eqi\}$, $i = 1, 2, ..., 6$) halmazát. A kimenő élek értéke a csúcshoz hozzá tartozó taszk működési idejével egyezik meg.
	Termék-csúcs: a gyártási folyamat során létrejövő termék.

2.8. ábra. Az S-gráf leírás építő elemei

Az S-gráf matematikai leírása a következő. Egy irányított G gráfot megadhatunk az $(\mathcal{N}, \mathcal{A})$ párral, ahol az $\mathcal{N}$ halmaz a csúcsok véges halmaza, az $\mathcal{A} \subseteq N \times N$ halmaz pedig az élek halmaza. Az S-gráf olyan irányított gráf, melynek két típusú éle van, az $\mathcal{A}1$ és az $\mathcal{A}2$ halmazbeli élek. Az $\mathcal{A}1 \subseteq N \times N$ halmaz a gráf úgynevezett recept-éleit (recipe-arc), az $\mathcal{A}2 \subseteq N \times N$ halmaz a gráf ütemezési-éleit (schedule-arc) tartalmazza, továbbá teljesül, hogy $\mathcal{A}1 \cap \mathcal{A}2 = \emptyset$. Bármely $(i, j) \in \mathcal{A}1 \cup \mathcal{A}2$ élhez tartozik egy nemnegatív $c(i, j)$ érték, az él súlya. Tehát a G S-gráf megadható egy $(\mathcal{N}, \mathcal{A}1, \mathcal{A}2)$ hármassal. A 2.9 ábrán láthatunk példát az S-gráf leírásra.

Egy ütemezést megvalósíthatónak nevezünk, ha az ütemezés alapján a gyártási folyamat végrehajtható. A megvalósítható ütemezés (feasible schedule) gondoskodik



2.9. ábra. Az S-gráf leírást szemléltető példa

a köztes anyagok tárolási stratégiának megfelelő kezeléséről, biztosítja, hogy a berendezések egy időben csak egy tevékenységet hajtsanak végre, minden tevékenységet végrehajt egy megfelelő berendezés, továbbá az ütemezés figyelembe veszi a berendezések új tevékenység elkezdéséhez szükséges váltási idejét és a tevékenységek között fennálló precedenciákat.

A jelen alfejezetben bemutatott S-gráf módszertan ábrázolási módja is és az általa megoldható feladatok is jelentős különbséget mutatnak a 2.2 fejezetben bemutatásra kerülő PNS módszertannal szemben. Alapvetően az S-gráf módszertan a szakaszos folyamatok optimális ütemezésének meghatározására lett bevezetve, míg a PNS módszertan segítségével a valamilyen szempont szerint optimális folyamathálózat struktúra határozható meg. Továbbá az STN és az RTN leírásokhoz hasonlóan az S-gráf sem képes kezelni azt az esetet, ha egy tevékenységet egyidőben két berendezés összedolgozva végez el, és ekkor a feladatmegosztástól függ a szükséges idő. Tehát a különbség abból is adódik, hogy a tevékenységek méretét nem képes változtatni még ha arra szükség is lenne, míg a P-gráf módszertan lehetőséget biztosít erre.

2.2. A folyamathálózat-szintézis kombinatorikus alapjai

Az ipari gyártási, termelési folyamatok tervezésekor gyakran felmerülő probléma, hogy a rendelkezésünkre álló bizonyos erőforrásokból és aktivitásokból előállítsuk az előírt

végcélokat valamely optimális folyamatstruktúra alapján. Az 1970-es években Fan és Friedler professzorok egy strukturális tulajdonságokra épülő technikát dolgoztak ki folyamathálózat-szintézis (Process Network Shynthesis, PNS) néven ([43], [48]). Szerepe kritikus az anyag- és energiafogyasztás, illetve a környezetre gyakorolt negatív hatások csökkentésében, ezáltal pedig a nyereségesség növelésében. A Siirola által publikált példa is szemlélteti, hogy a hatékony folyamatszintézis a költséget 35, az energiafogyasztást pedig akár 50%-kal is csökkentheti ([98]).

A PNS-t vegyipari termelési folyamatok modellezésére és optimalizálására dolgozták ki, ezért az eredeti terminológia leginkább erre a területre alkalmazható. Mivel egy robusztus, a különböző típusú folyamatoptimalizációs problémákra jól adaptálható módszertanról van szó – amely a jelen kutatás alapjául is szolgál – ezért célszerűbb egy általánosabb terminológia bevezetése. Ebben a műveleti egységeket aktivitásoknak vagy tevékenységeknek az anyag típusú csomópontokat entitásoknak ezen belül is a nyersanyagokat erőforrásnak vagy kiindulási feltételeknek a termékeket végcéloknak és a köztes anyagokat vagy termékeket pedig köztes entitásoknak, előfeltételnek vagy következménynek nevezzük. A továbbiakban ez az újonnan bevezetett terminológia fog szerepelni mind a már korábban kidolgozott alapok, mind az újonnan született eredmények terén.

Ideális esetben egy folyamat struktúráját, és a folyamatot alkotó aktivitások konfigurációit egyidejűleg lehetne megtervezni és szintetizálni, mivel a teljesítményük kihatással van egymásra. Ez azonban a gyakorlatban rendkívül nehéz, mondhatni lehetetlen, köszönhetően a probléma duális, egyidejűleg folytonos és diszkrét természetének. Az utóbbi a feladat kombinatorikus komplexitásához vezet, ami a probléma optimális megoldásának megtalálását jelentősen megnehezíti. Emiatt a folyamatszintézisnek három fázisát különböztetjük meg: a makroszkopikus, mezoszkopikus és a mikroszkopikus fázist. A makroszkopikus a legmagasabb absztrakciós szint, itt a legnagyobb a modell által átfogott terület, ugyanakkor ez a legkevésbé részletes. A mikroszkopikus modell a legkisebb absztrakciós szint. Egy mikroszkopikus modell a rendszernek általában csak kis részletét emeli ki, de azt nagyon részletesen írja le. Ilyen modellek adják meg például egy berendezés működését, a benne lejátszódó fizikai, kémiai vagy biológiai folyamatokat. A mezoszkopikus szint léptéke a makró és a

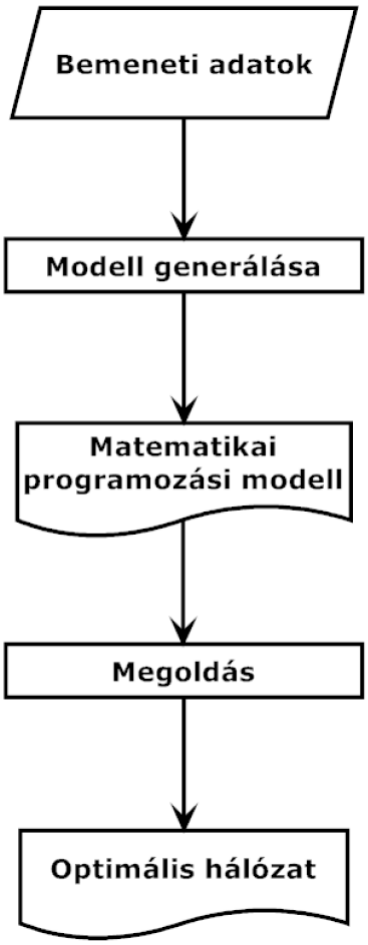
mikró közé esik. Itt található a technológiai lépések kapcsolatainak részletezése, vagy a koncepcionális modell bizonyos szempontú kifejtése. Ahogyan az a nevükből is következik, a három fázist az alapján különböztetjük meg, hogy milyen részletességgel foglalkoznak a folyamat megtervezésével. A részrendszerek, a funkcionális egységek összekapcsolása, azaz a rendszer szintézise a makroszkopikus fázisban megy végbe, ezért a folyamatszintézis szempontjából ez a fázis a legjelentősebb.

A folyamatszintézis feladatok megoldására kidolgozott módszereket két nagy csoportba osztjuk, mégpedig a heurisztikus és az algoritmikus azaz matematikai programozáson alapuló módszerekre. Léteznek úgynevezett hibrid módszerek is, amelyek heurisztikus szabályok mellett egyidejűleg támaszkodnak a matematikai programozásra is.

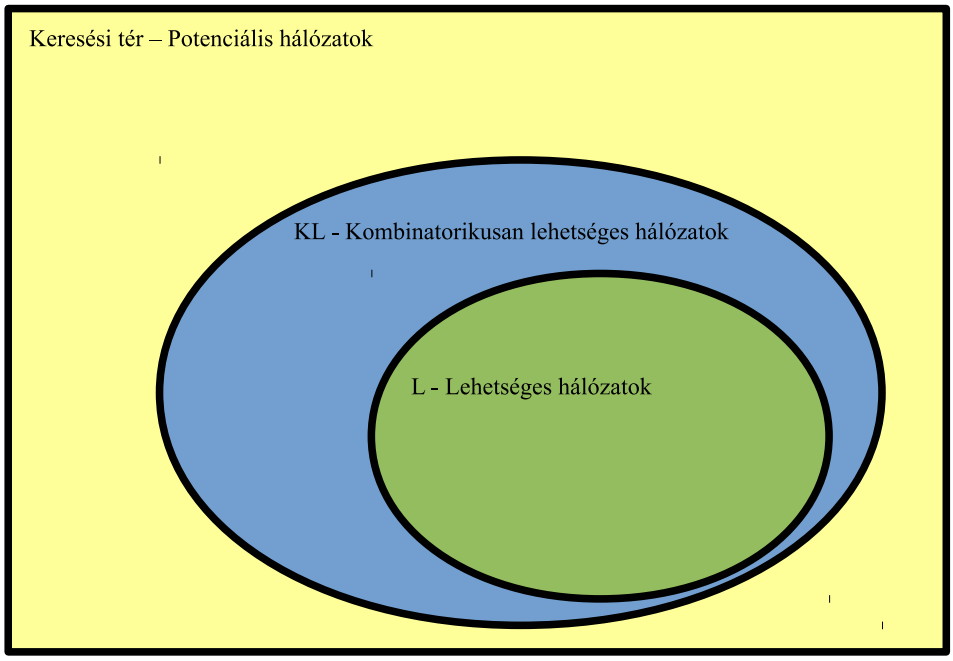
A heurisztikus módszerek megvalósítása általában egyszerű, még nagy feladatok esetében is, azonban természetüknél fogva csak lokálisan hatékonyak. Ennek oka, hogy az emberi tapasztalatok, melyeken a heurisztikák szabályai alapulnak, véges, és gyakran korlátozott, limitált számú megfigyelésből erednek. Következésképpen a heurisztikus módszerek önmagukban gyakran alkalmatlanok a globális, vagy közel globális optimális megoldások megtalálására ([37]).

A heurisztikus módszerekkel szemben, a meglévő, hagyományos algoritmikus módszerek, melyek főbb lépései az 2.10 ábrán láthatóak, csak viszonylag kis, illetve mérsékelt méretű feladatok kezelésére alkalmasak. Ezek az algoritmikus módszerek csak akkor lesznek megfelelően precízek, ha az általuk használt matematikai programozási modelleket explicit módon meg tudjuk konstruálni.

Ezen módszerek esetében a feladat definíciója mindig megadja a rendelkezésre álló erőforrásokat, a végcélokat valamint a felhasználható aktivitásokat. A különböző kapcsolódó paraméterekre – azaz költségekre, árakra, végrehajtási időkre illetve anyagegyensúly feltételekre – írjuk fel a matematikai programozási modelleket. Ezeket a modelleket általában „félig-meddig” korábbi tapasztalatokra alapozva adják meg, folytonos függvények segítségével. Mindemellett ezek a hagyományos módszerek nem tudják közvetlenül és szisztematikusan, azaz algoritmikusan képezni a célfüggvényt és a kapcsolódó feltételrendszert, amely megfelelően kifejezné a folyamatok hálózatstruktúráját. Nem adnak módszert arra sem, hogy hogyan lehetne algoritmikusan



2.10. ábra. A hagyományos algoritmikus módszerek főbb lépései



2.11. ábra. A Keresési tér csökkentése

elkészíteni egy olyan struktúráját a folyamatnak, amely minden lehetséges hálózatot redundancia nélkül tartalmaz. Ez azért kritikus, mert a matematikai programozás egyrészt csak olyan eredményt adhat, ami része a matematikai modellnek, másrészt használhatatlanul lelassulhat, ha a modell indokolatlanul nagy.

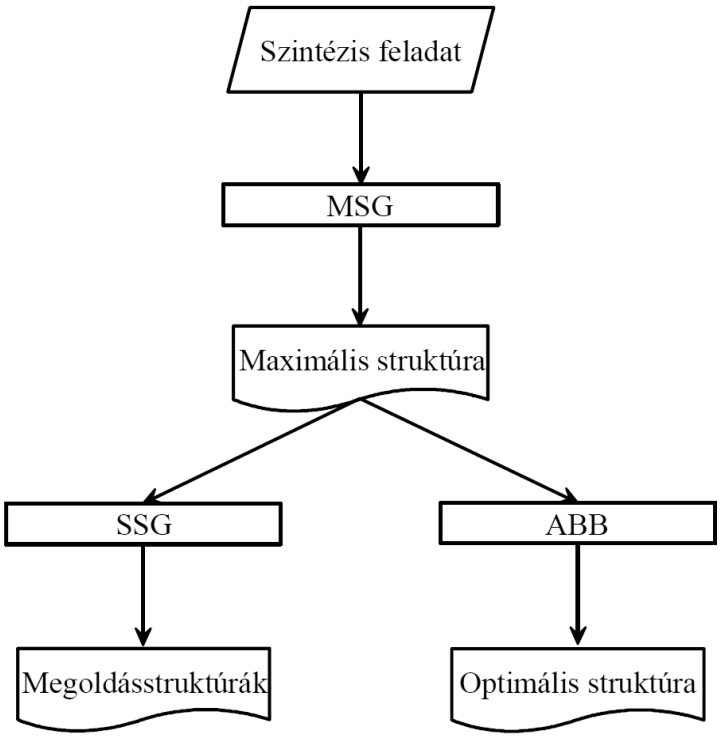
A folyamatszintézis komplexitását a probléma kettős, egyidejűleg folytonos és diszkrét természete okozza. Az utóbbinak köszönhetően a probléma komplexitása exponenciálisan növekszik a rendelkezésre álló aktivitások számával, amit jelöljön n, mivel az optimális hálózatot 2^{n-1} lehetséges alternatíva közül kellene meghatározni mindaddig, amíg nem rendelkezünk olyan tudással a hálózat struktúrájáról, ami alapján helytelen alternatívák eltávolításával ezt a számot csökkenteni lehetne. Olyan robusztus döntéstámogatási rendszerekre van tehát szükség a folyamatszintézis feladatok kombinatorikus komplexitásának megfelelő kezelésére, amelyek matematikailag szigorúak, lehetőleg axiomatikusak, és számítógépekre hatékonyan implementálhatóak.

Mindez a kombinatorika egy jól kidolgozott ágának, a gráfelmélet eredményeinek

felhasználásával valósítható meg. Eredményül egy gráfelméletben gyökerező, algoritmikus módszert kapunk, amely a későbbiekben részletesebben is bemutatásra kerül. A módszer a gráfok egy speciális osztályán alapszik, amely segítségével a folyamatok struktúrája egyértelműen reprezentálható, így lehetőség nyílik a gyakorlatban is megvalósítható folyamatokra jellemző kombinatorikus tulajdonságok kihasználására. Matematikailag az ilyen kombinatorikus tulajdonságok axiómaként fogalmazhatóak meg, amelyek számos algoritmus alapjául szolgálnak ([43], [44], [45], [46]). A 2.11 ábrán látható, hogyan csökkenti a módszer a keresési teret az optimális megoldás keresése során. A keresési tér, amely az összes lehetséges hálózatot tartalmazza, leszűkül azokra a hálózatokra, amelyek kielégítik az axiómákat, azaz az úgynevezett kombinatorikusan lehetséges hálózatokra, amit az ábrán KL jelöl. A két halmaz mérete között több nagyságrendbeli eltérés is lehet, amely a megoldáshoz igényelt számítási kapacitás drasztikus csökkenéséhez vezethet. A probléma duális, folytonos és kombinatorikus (egész) természete miatt egy folyamatszintézis feladat matematikai programozási modelljéül vagy egy MILP, vagy egy MINLP szolgál ([30], [76], [33], [76], [86], [28], [41], [87], [71], [21], [112], [108], [39]). A továbbiakban bemutatásra kerül a már említett, gráfelméleti alapokon nyugvó, algoritmikus módszer, amivel még a nagy komplexitású folyamatok is hatékonyan kezelhetőek és modellezhetőek folyamatszintézis feladatként.

2.2.1. A P-gráf

A PNS-feladat ábrázolásához elengedhetetlen volt egy új jelölés rendszer bevezetése, mivel az aktivitások és az entitások közötti reláció szemléltetése egy hagyományos gráf csúcsaival és éleivel nem lehetséges ([43]). A probléma megoldására született meg a P-gráf (Process Graph - Folyamat gráf, P-graph), amely egy irányított páros gráf, és segítségével lehetőség nyílik a lehetséges megoldásstruktúrák kombinatorikus tulajdonságainak kihasználására, ezáltal pedig a nagyméretű feladatok optimumának meghatározására is. Egy gráfot párosnak nevezünk, ha csúcsai két diszjunkt halmazba sorolhatók, és az ugyanabba a halmazba tartozó csúcsok nem lehetnek szomszédosak. A P-gráf módszertant az 1990-es évek elején Friedler és szerzőtársai dolgozták ki



2.12. ábra. A P-gráf módszertan főbb lépései

komplex vegyipari termelőrendszerek modellezésére és optimalizálására ([44], [45], [46]). A módszertan főbb lépései a 2.12 ábrán láthatóak. A következőkben a modell megadásához szükséges alapfogalmak definiálására kerül sor.

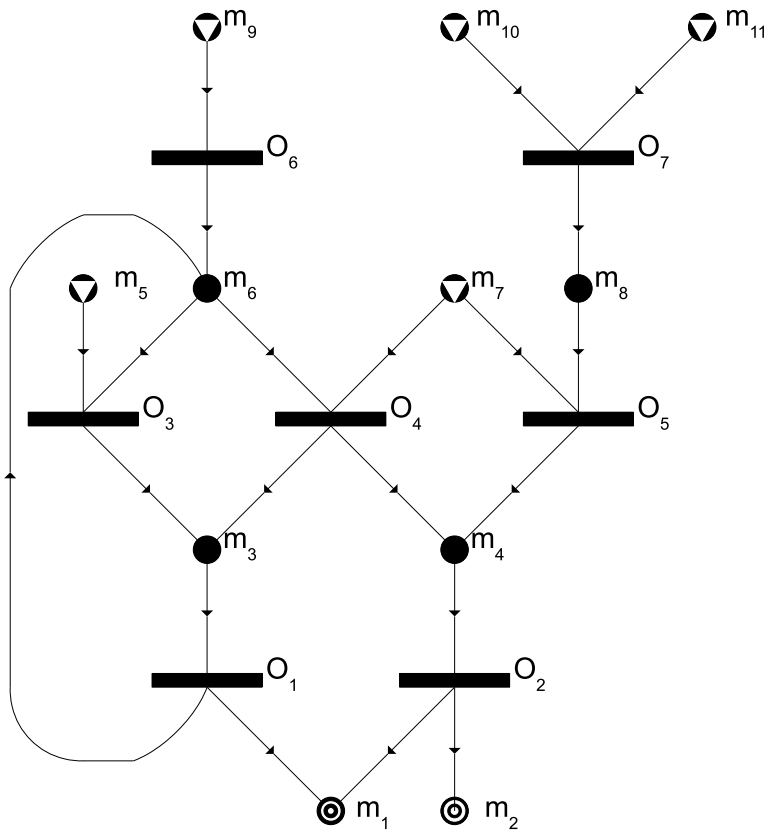
Definíció 2.2.1 *Legyen az $\mathcal{M}$ az entitások, valamint az $\mathcal{O} \subseteq \wp(\mathcal{M}) \times \wp(\mathcal{M})$ az aktivitások véges és nem üres halmaza, ahol $\wp(\mathcal{M})$ jelöli $\mathcal{M}$ hatványhalmazát. Az $(\mathcal{M}, \mathcal{O})$ párt P-gráfnak nevezzük. A csúcsok halmaza $(\mathcal{M} \cup \mathcal{O})$, az élek halmaza $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, ahol*

$$\mathcal{A}_1 = \{(x, Y) : Y = (\alpha, \beta) \in \mathcal{O}, x \in \alpha\},$$

$$\mathcal{A}_2 = \{(Y, z) : Y = (\alpha, \beta) \in \mathcal{O}, z \in \beta\}.$$

Ha $(\alpha, \beta) \in \mathcal{O}$, akkor α a bemeneti entitások halmaza, β pedig a kimeneti entitásoké.

Definíció 2.2.2 *Az $(\mathcal{M}, \mathcal{O})$ P-gráf részgráfjának nevezzük az $(\mathfrak{m}, \mathfrak{o})$ gráfot, ha az $\mathfrak{m} \subseteq \mathcal{M}$ és $\mathfrak{o} \subseteq \mathcal{O}$ feltételek teljesülnek.*



2.13. ábra. Egy termelési feladat P-gráfja

Megjegyzés 2.2.1 *A dolgozat további részében egy kitöltött téglalappal jelölt csúcsra aktivitásként, egy kitöltött körre pedig entitás jellegű csomópontként (röviden entitásként) hivatkozunk.*

Példa 2.2.1 *A 2.13 ábra egy P-gráfot reprezentál, mely egy olyan termelési feladatot ír le, ahol az m_1 anyagot mint végcélt szeretnénk elérni a rendelkezésünkre álló erőforrások $(m_5, m_7, m_9, m_{10}, m_{11})$ felhasználásával. A termelés során keletkezhetnek köztes entitások (m_3, m_4, m_6, m_8) vagy akár mellékcélok is (ilyen például az m_2 csúcspont által reprezentált entitás).*

Ahogy az korábban bemutatásra került, a P-gráf megalkotásakor a gráf tartalmazhat olyan csúcsokat és éleket, melyek a cél elérése szempontjából nem lényegesek (ilyen csúcs lehet a 2.13 ábrán az m_2 entitás, mely elhagyható, hiszen esetünkben nem

a végcélt reprezentálja). Ezért fontos lépés lehet – már a feladat méretének csökkentése miatt is – a kiindulási feladat egy olyan részgráfjának meghatározása, mely csak azon csúcsokat tartalmazza, melyek a feladat megoldása során, mint lehetséges csúcspontok jelennek meg. Miért lehet nagyon fontos ez a lépés számunkra a fentieken túl? Az üzleti folyamat P-gráf leírásából egyértelműen azonosíthatóak lesznek azon erőforrások és aktivitások, amelyek nem járulnak hozzá sem közvetlen, sem közvetett módon a profit növekedéséhez, tehát „felesleges" folyamatok, de ami még fontosabb, a vizsgált vállalkozás, szervezet számára is erőforrás-pazarló tevékenységek. Ezen tevékenységek azonosítása kulcsfontosságú a folyamatok kezelése és vizsgálata során.

Az alábbi definícióval és a 2.2.3. fejezettel bevezetésre kerül a kombinatorikusan lehetséges struktúra fogalma, amely alapvető fontosságú lesz a későbbiek során ([44], [45], [46], [47], [43]).

Definíció 2.2.3 *Legyen $\mathcal{P} \subseteq \mathcal{M}$ és $\mathcal{R} \subseteq \mathcal{M}$ a végcélok illetve a erőforrások egymástól diszjunkt halmaza. Ekkor a $(\mathcal{P}, \mathcal{R}, \mathcal{O})$ hármast a vizsgált PNS-feladat strukturális modelljének nevezzük.*

Példa 2.2.2 *A 2.13 ábrán látható P-gráf strukturális modellje a következő: $(\mathcal{P}, \mathcal{R}, \mathcal{O})$, ahol $\mathcal{P} = \{m_1\}$ a végcél(ok), $\mathcal{R} = \{m_5, m_7, m_9, m_{10}, m_{11}\}$ az erőforrások halmaza, továbbá*

$$\begin{aligned}\mathcal{O} = \{&(\{m_3\}, \{m_1, m_6\}),\\ &(\{m_4\}, \{m_1, m_2\}),\\ &(\{m_5, m_6\}, \{m_3\}),\\ &(\{m_6, m_7\}, \{m_3, m_4\}),\\ &(\{m_7, m_8\}, \{m_4\}),\\ &(\{m_9\}, \{m_6\}),\\ &(\{m_{10}, m_{11}\}, \{m_8\})\}.\end{aligned}$$

2.2.2. Strukturális tulajdonságokat leíró leképezések

Ha (α, β) egy eleme a $\mathcal{O}$ halmaznak, akkor az (α, β) aktivitás bemeneti entitásainak azaz előfeltételeinek halmazát az α jelöli és a kimeneti entitásainak, azaz következményeinek halmazát pedig a β ahogy az Friedler és szerzőtársai munkáságaiban is látható ([44], [47], [48]). Egy $(\mathcal{m}, \mathcal{o})$ P-gráfra,

- $\psi^-(\mathcal{o})$ jelöli egy folyamathálózat struktúra entitásainak egy olyan halmazát, amelynek mindegyike legalább egy az $\mathcal{o}$ halmazba tartozó tevékenység előfeltétele. Formálisan:
$$\psi^-(\mathcal{o}) = \bigcup_{(\alpha,\beta)\in\mathcal{o}} \alpha$$

- $\psi^+(\mathcal{o})$ jelöli egy folyamathálózat struktúra entitásainak egy olyan halmazát, amelynek mindegyike legalább egy az $\mathcal{o}$ halmazba tartozó tevékenység következménye. Formálisan:
$$\psi^+(\mathcal{o}) = \bigcup_{(\alpha,\beta)\in\mathcal{o}} \beta.$$

- $\psi(\mathcal{o})$ jelöli továbbá az $\mathcal{o}$ halmazban szereplő tevékenységek előfeltétel, vagy következmény entitásainak halmazát. Formálisan:
$$\psi(\mathcal{o}) = \psi^-(\mathcal{o}) \cup \psi^+(\mathcal{o})$$

- $\varphi^-(\mathcal{m})$ jelöli egy folyamathálózat struktúra tevékenységeinek egy olyan halmazát, amely tevékenységek mindegyikének legalább egy az $\mathcal{m}$ halmazban található entitás a következménye. Formálisan:
$$\varphi^-(\mathcal{m}) = \{(\alpha, \beta) \in \mathcal{o} : \beta \cap \mathcal{m} \neq \emptyset\}$$

- $\varphi^+(\mathcal{m})$ jelöli egy folyamathálózat struktúra tevékenységeinek egy olyan halmazát, amely tevékenységek mindegyikének legalább egy az $\mathcal{m}$ halmazban található entitás az előfeltétele. Formálisan:
$$\varphi^+(\mathcal{m}) = \{(\alpha, \beta) \in \mathcal{o} : \alpha \cap \mathcal{m} \neq \emptyset\}$$

- $\varphi(m)$ jelöli azon aktivitások halmazát, amelyeknek az m halmazban szereplő entitások az előfeltételei vagy következményei. Formálisan:

$$\varphi(m) = \varphi^-(m) \cup \varphi^+(m)$$

- Legyen m' és m'' két entitás halmaz és az o egy aktivitás halmaz. Ekkor a $\Rightarrow_o$ művelet a következőképpen definiálható:

$$m' \Rightarrow_o m'' \text{ akkor és csak akkor, ha } m'' = \psi^+(\varphi^+(m') \cap o)$$

  Vagyis a művelet az entitások egy olyan m'' halmazát eredményezi, amely az o halmazban található azon aktivitások következménye, amelyeknek legalább egy előfeltétele megtalálható az m' halmazban. A $\Rightarrow_o$ művelet reflexív és tranzitív lezártján a $\Rightarrow_o^*$ műveletet értjük.

2.2.3. A kombinatorikusan lehetséges struktúrák szükséges axiómái

Bizonyos kombinatorikus tulajdonságok a természetes velejárói minden egyes lehetséges folyamat struktúrának. Friedler és szerzőtársai matematikailag ezeket a tulajdonságokat axiómagyűjteményként fejezték ki ([44], [45], [46], [43]).

Definíció 2.2.4 *Legyen adott egy PNS-feladat $(\mathcal{P}, \mathcal{R}, \mathcal{O})$ strukturális modellje és legyen (m, o) P-gráf az $(\mathcal{M}, \mathcal{O})$ P-gráf részgráfja. Az (m, o) részgráfot kombinatorikusan lehetséges struktúrának nevezzük, ha teljesül az alábbi öt feltétel mindegyike:*

(S1) Minden végcél megtalálható a gráfban, azaz $\mathcal{P} \subseteq m$;

(S2) Egy entitás típusú csúcshoz pontosan akkor nem tartozik bemenő él, ha az erőforrást reprezentál, azaz $m \setminus \psi^+(o) = m \cap \mathcal{R}$.

(S3) Minden a struktúrában szereplő tevékenység a szintézis feladatban definiált, azaz $o \subseteq \mathcal{O}$.

(S4) Minden a struktúrában szereplő tevékenységből vezet út legalább egy elérendő végcélhoz, vagyis nincs izolált csúcs a részgráfban, azaz $\forall o' \in o, \exists m' \in m$ *úgy, hogy* $\psi^+(o') \Rightarrow_o^* m'$ *és* $m' \cap \mathcal{P} \neq \emptyset$.

(S5) Ha egy $x \in m$ *entitás része a struktúrának, akkor létezik a struktúrában olyan tevékenység, amelynek az* x *entitás előfeltétele (bemenete) vagy következménye (kimenete), azaz* $m \subseteq \psi(o)$.

2.2.4. Az MSG algoritmus

Mivel a folyamatszintézis feladatok megoldásai exponenciális nagyságrendűek, így biztosan nincs algoritmus, ami ezeket a megoldásokat exponenciálisnál kisebb lépésszámmal leszámlálná. Így a hatékonyság növelésének kulcsa – a lehetséges folyamatok szükséges tulajdonságait felhasználva – úgy csökkenteni a keresési teret, hogy közben egyetlen megoldást se veszítsünk el. A keresési tér csökkentése pedig ebben az esetben a figyelembe veendő tevékenységek halmazának szűkítését jelenti. A 2.14 ábrán látható és most ismertetésre kerülő, MSG (Maximal Structure Generator) algoritmus bemenete a folyamatok P-gráf leírása, kimenete pedig annak maximális struktúrája. Ez egy bizonyítottan gyors, polinom lépésszámú algoritmus ([45], [46]).

Az MSG algoritmus a feladat megoldása kezdetén szűri ki azokat az tevékenységeket, amelyek biztosan nem szerepelhetnek egyetlen kombinatorikusan lehetséges megoldásban sem. Az algoritmus futásának – amely két fázisra bontható, a redukciós és az építési fázisra – eredményeként megkapjuk a feladat maximális struktúráját.

A első, lebontó (redukáló) jellegű rész kiszűri a már ismertetett axiómáknak nem megfelelő tulajdonságú tevékenységeket, illetve csak az axiómák előfeltételeként vagy következményeként szereplő entitásokat. A második, építő rész pedig azért felelős, hogy csak a feltétlenül szükséges tevékenységek és azokkal együtt az entitások egy meghatározott része legyenek a modell részei.

```
01. function MSG((𝓜, 𝒪):PNS problem): (m, o):P-graph
02. begin
03.  o := 𝒪 \ φ⁻(ℛ);
04.  r := ψ⁻(o) \ ψ⁺(o) \ ℛ;
05.  comment: reduction part
06.  while r ≠ ∅ do
07.        o := o \ φ⁺(r);
08.        r := ψ⁻(o) \ ψ⁺(o) \ ℛ;
09.  end while
10.  if 𝒫 ⊄ ψ⁺(o) then
11.        return;
12.  end if
13.  comment: construction part
14.  p := 𝒫; o′ := ∅; Δo := ∅;
15.  repeat
16.        Δo := φ⁻(p) ∩ o \ o′;
17.        o′ := o′ ∪ Δo;
18.        p := ψ⁻(Δo);
19.  until Δo ≠ ∅
20.  return(ψ(o′), o′);
21. end;
```

2.14. ábra. Az MSG algoritmus

2.2.5. Az SSG algoritmus

Vannak olyan esetek, amikor nemcsak valamilyen szempont szerinti optimális megoldást szeretnénk megkeresni, hanem az összes megoldás előállítása a cél. Minden szervezet számára fontos, hogy az egyes folyamataikra vonatkozó összes alternatív lehetőséget lássák maguk előtt. Ennek alapján az adott szervezet az általa meghatározott cél érdekében döntést tud hozni arról, hogy a matematikailag optimális utat választja-e vagy más, hivatalosan még meg nem határozott szempontok szerint dönt egy megvalósítás mellett. Az alternatív lehetőségek felsorolásának egyik célja tehát a lehetséges fejlesztési lehetőségek megmutatása, illetve a matematikai modellbe nem adaptálható szubjektív szempontok kielégítése a folyamatok elemzéséhez.

Az 2.15 ábrán látható SSG (Super Structure Generator) algoritmus megértéséhez szükséges néhány további paraméter tisztázása. A p jelöli az entitások azon halmazát amelyekről még el kell döntetni miként állnak elő, a p^+ pedig azokat az entitásokat amelyekről még nem dőlt el miként keletkeznek. Az o^+ jelöli a tevékenységek azon halmazát amelyek már be lettek választva a o^- pedig azokat amik már ki lettek zárva a struktúra generálás során ([44], [47]).

Az alternatív lehetőségek felsorolásának másik előnye a folyamatban bekövetkező hibaesemény hatásakor mutatkozik meg. Egy hibalehetőség kiküszöbölése vagy a hibaesemény gyors elhárítása lehetségessé válhat egy alternatív út, vagy lehetőség kiválasztásával és az eredeti folyamatba való beillesztésével.

Ebben az alfejezetben bemutatásra kerül az SSG algoritmus, amely az összes kombinatorikusan lehetséges struktúra generálását megvalósító algoritmus és amely a 2.15 ábrán látható. Az SSG algoritmus hatékonysága abban áll, hogy csak megoldásstruktúrákat eredményez, mégpedig mindegyiket pontosan egyszer.

Az eljárás maga egy rekurzív magra épül. Látható, hogy az algoritmus alulról építkezik: az összes kombinatorikusan lehetséges megoldásstruktúra egy ciklus segítségével áll elő, a különböző döntési leképezések hatására, cikluson belül olyan rekurzív függvényhívást használva, amely mindig végcéloktól indul, lépésenként az őket eredményező tevékenységeket beválasztva, amíg el nem jut az erőforrásokig. Az SSG előállítja tehát az összes kombinatorikusan lehetséges esetet, ami megfelelő alapot ad

```
01. function SSG((m, o):P-graph):set of solution structures
02. begin
03.  solutions := ∅;
04.  subproblems := {(𝒫, ∅, ∅, (O \ o))};
05.  while subproblems ≠ ∅ do
06.        let (p, p⁺, o⁺, o⁻) ∈ subproblems;
07.        subproblems := subproblems \ (p, p⁺, o⁺, o⁻);
08.        if p = ∅ then
09.              solutions := solutions ∪ {(ψ(o⁺), o⁺)};
10.        else
11.              BranchSSG((p, p⁺, o⁺, o⁻), subproblems);
12.        end if
13.  end while
14.  return solutions;
15. end;
16.
17. function BranchSSG((p, p⁺, o⁺, o⁻) : subproblem, subproblems: set of
                        subproblems)
18. begin
19.  Let x ∈ 𝒫;
20.  o_x := φ⁻({x}) \ o⁻; o_xb := (o_x ∩o⁺);
21.  𝒞 := ℘(o_x \ o_xb);
22.  if o_xb := ∅ then
23.        𝒞 := 𝒞 \ {∅};
24.  end if
25.  for all c ∈ 𝒞 do
26.        subproblems := subproblems∪ ((p ∪ ψ⁻(c)) \ p⁺ \ {x} \ ℛ, p⁺ ∪ {x}, o⁺ ∪ {c},
                        o⁻ ∪ (o_x \ o_xb \ c) ));
28.  end for
29. end;
```

2.15. ábra. Az SSG algoritmus

üzleti folyamatok előállítására is.

2.2.6. Az ABB algoritmus

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

A vegyes-egész programozási feladatoknak a megoldására használhatóak az általános korlátozás és szétválasztás (branch-and-bound, B&B) típusú módszerek ([69], [34], [57], [34], [8], [18], [68], [9], , [87]). Az ilyen típusú algoritmusok két fő részből tevődnek össze: egy szétválasztó és egy korlátozó függvényből. Bár a feladat optimális megoldása ezekkel a módszerekkel is meghatározható, hatékonyságuk még tovább javítható, mivel a megoldás keresése során nem veszik figyelembe a szintézis feladatok speciális tulajdonságait.

A PNS probléma NP teljességéből következik, hogy nem várható polinom idejű megoldás – a B&B algoritmusok is exponenciális idejűek – ám megfelelő szétválasztási, illetve korlátozó függvényekkel és egyéb eszközökkel gyorsíthatóak, ami azt jelenti, hogy empirikus vizsgáltok során a futási idő jóval kevesebb mint eredetileg.

A P-gráf módszertan az optimális megoldás meghatározására ennek megfelelően egy speciális korlátozás és szétválasztás típusú algoritmust, az ABB-t (Accelerated Branch-and-Bound) használja ([48], [59]). Az ABB algoritmus a valamilyen szempontból optimális megoldás megtalálására alkalmas. Maga az algoritmus az SSG döntési leképezésein alapul, amelynek az az előnye, hogy az ABB csak a kombinatorikusan lehetséges struktúrákat vizsgálja az optimális megoldás keresése során, azonban különbözik abban, hogy a döntés előtt az algoritmus egy korlátszámítási eljárást hajt végre, amely a részproblémákhoz egy alsó korlátot rendel. Ez az alsó korlát alapján kizárható részproblémákat ismer fel, így az ABB eljárás az SSG által bejárt fa csak egy részét generálja. Ezen a részfán halad folyamatosan tovább mindaddig míg el nem éri a levelet. Az algoritmus annál hatékonyabb, minél kisebb a bejárt fa, amit

döntően a korlátozási eljárásban meghatározott alsó korlát élessége határoz meg.

Az algoritmus alkalmazhat egyéb kombinatorikus gyorsításokat is, mint például az úgynevezett neutrális kiterjesztést, amely meggyorsítja az algoritmust abban az esetben, amennyiben egy entitás esetében teljesen egyértelmű, hogy milyen tevékenység eredményezi azt az eddig meghatározott döntések után. Az ABB algoritmus részletesebb tárgyalása a 3.4 fejezetben történik.

3. fejezet

PNS feladat fix részt tartalmazó lineáris költség függvénnyel

A most következő fejezetben a dolgozat alapfeladat formális megfogalmazása, azaz a paraméteres PNS feladat fix részt tartalmazó lineáris költségfüggvényű változata kerül bemutatásra. A dolgozat további részeiben a jelen fejezetben bemutatásra kerülő alapfeladat formális leírását alapul véve keressük majd azokat a megoldásokat, amelyek a 4. fejezetben bemutatásra kerülő kölcsönösen kizáró feltételeket, az 5. fejezetben ismertetendő megbízhatósági korlátokat és a 6. fejezetben bemutatandó időkorlátokat teljesítik.

3.1. A relaxált feladat

Egy vegyes egész matematikai programozási feladatot (MILP) úgy oldunk meg, hogy tekintjük annak egy folytonos változatát, amely nem korlátozza, azt hogy az egyes változóknak egészeknek kell lennie. Például bináris változók esetén csak annyit kötünk meg, hogy a változó értéke a [0,1] intervallumban van. Ezt hívjuk relaxált feladatnak. Ezzel egy az eredetinél bővebb megengedett tartományt kapunk, amely tartalmazza részeként az eredeti megengedett tartományt is. Ezen bővebb halmazon optimalizálva biztosan nem kaphatunk az eredetinél rosszabb megoldást, így ennek

optimuma – minimalizálás esetén – alsó korlátként szolgálhat az eredeti feladat optimumának becsléséhez. Az optimalizálás során, ha a relaxált feladat megoldásában minden, eredetileg egész változó értéke a relaxált feladat megoldásában is egész lett, akkor megkaptuk az eredetei feladat megoldását. Ha valamely eredetileg egész változó értékére a relaxált feladat megoldásában nem egész értéket kapunk, akkor egy ilyen változót kiválasztva, a feladatot több részre bontjuk, és a lehetséges egész értékek mindegyikére külön esetet tekintünk. Az eseteket addig daraboljuk tovább, míg a folyamat során lerögzített egész értékek mellett már, az összes eredetileg egész változó értéke a relaxált feladat megoldásában is egésszé válik.

3.2. Modell paraméterek

Egy matematikai optimalizálási feladatnak tartalmaznia kell a logikai megkötéseket és a célfüggvényt. Folyamat szintézis feladat esetén az anyagegyensúly, az erőforrások elérhetősége és az elérendő végcélok mennyiségének alsó korlátja alkotják a megkötéseket. Ezek a megkötések a következőkben kerülnek kifejtésre.

Minden egyes m_j végcél bruttó alsó korlátja Lp_j nagyobb mint nulla, és ez az alsó korlát egyenlő nullával minden más entitásra, azaz

$$Lp_j = \begin{cases} > 0, & \forall m_j \in \mathcal{P} \\ 0, & \textit{különben} \end{cases}. \tag{3.2.1}$$

Minden egyes m_j erőforrás bruttó felső korlátja Up_j egyenlő nullával, és ez a felső korlát nagyobb vagy egyenlő az Lp_j alsó korláttal minden más entitás esetében, azaz

$$Up_j = \begin{cases} 0, & \forall m_j \in \mathcal{R} \\ \geq Lp_j, & \textit{különben} \end{cases}. \tag{3.2.2}$$

Minden egyes m_j erőforrás felhasználásának bruttó felső korlátja Uc_j nagyobb nullánál, és ez a felső korlát egyenlő nullával minden más entitás esetében, azaz

$$Uc_j = \begin{cases} > 0, & \forall m_j \in \mathcal{R} \\ 0, & \textit{máskülönben} \end{cases} . \tag{3.2.3}$$

Az összes tevékenység kapacitásánaku_i felső korlátja és az összes erőforrás és végcél cm_j költsége szintén adott. Egy tevékenység költsége a kapacitásának egy fix résszel vett lineáris függvényével becsülhető meg. Minden tevékenység költségfüggvénye esetén a cp_i arányossági konstans és a cf_i egész rész definiáltak. Az

$$L_j = \begin{cases} -\mathcal{U}c_j, & \forall m_j \in \mathcal{R} \\ \mathcal{L}p_j, & \textit{különben} \end{cases} \tag{3.2.4}$$

alsó korlát és az

$$U_j = Up_j \tag{3.2.5}$$

felső korlát jelöli az eredmény és az erőforrások felhasználása valamint a tevékenységek kimenete közötti egyensúlyt. Továbbá, jelölje $\mathbf{m} \subseteq \mathcal{M}$ az entitások halmazát és $\mathbf{o} \subseteq \mathcal{O}$ a tevékenységek halmazát az optimális struktúrában. Az entitások és tevékenységek közötti kapcsolatok az a_{ji} paraméterekkel adottak, amik az o_i tevékenység által eredményezett és felhasznált m_j entitás optimális mennyiségét jelölik.

3.2.1. Kiegészítő jelölések

Jelölje $\mathbf{o}^0$ azokat a tevékenységeket amelyekről még nem született döntés az optimális megoldás keresése során. Az $\mathbf{m}^*$ azokat a entitásokat, az $\mathbf{o}^*$ pedig azokat a tevékenységeket jelöli, amelyeket az optimális megoldás tartalmaz. Az x^* jelöli az aktivitások optimális mennyiségének vektorát a feladat szempontjából és z^* pedig a feladat célértékét.

Az $\widetilde{\mathbf{m}}$ jelöli azokat a entitásokat az $\widetilde{\mathbf{o}}$ pedig azokat a tevékenységeket, amelyeket a relaxált feladat megoldása tartalmaz. Az $\widetilde{x}$ reprezentálja a tevékenységek optimális mennyiségének vektorát a relaxált feladat szempontjából és $\widetilde{z}$ pedig a relaxált feladat

célértékét mutatja.

3.3. Feladat meghatározás

A cél, hogy meghatározzuk azt az $(\mathcal{m}^*, \mathcal{o}^*, \mathbf{x}^*, z^*)$ hálózatot, amely kielégíti a 3.3.1 - 3.3.6 egyenleteket és a z^* minimális, ahogy az a 3.3.7 egyenletben látható.

$$\mathcal{m}^* = \psi(\mathcal{o}^*), \text{ vagyis legyen } m^* \text{ az entitások egy olyan halmaza amely legalább egy } o^* \text{ halmazbeli tevékenység előzménye vagy következménye.} \tag{3.3.1}$$

$$\mathbf{x}^* = [x_1^*, x_2^*, ..., x_n^*]^T \tag{3.3.2}$$

$$0 < x_i^* \leq u_i \iff o_i^* \in \mathcal{o}^* \tag{3.3.3}$$

$$\forall m_j \in \mathcal{m}^* \cap \mathcal{R} : -Uc_j \leq \sum_{o_i \in \mathcal{o}^*} a_{ji} x_i^* \leq 0 \tag{3.3.4}$$

$$\forall m_j \in \mathcal{m}^* \cap \mathcal{P} : Lp_j \leq \sum_{o_i \in \mathcal{o}^*} a_{ji} x_i^* \leq Up_j \tag{3.3.5}$$

$$\forall m_j \in \mathcal{m}^* \setminus \mathcal{R} \setminus \mathcal{P} : 0 \leq \sum_{o_i \in \mathcal{o}^*} a_{ji} x_i^* \leq Up_j \tag{3.3.6}$$

$$z^* = \sum_{(\alpha_i, \beta_i) = o_i \in \mathcal{o}^*} (cf_i + x_i^*(cp_i - \sum_{m_j \in \alpha \cup \beta} a_{ij} cm_j)). \tag{3.3.7}$$

3.4. Az ABB algoritmus paraméteres PNS feladatként történő leírása

Az előző fejezetekben bevezetett kiegészítő paramétereket, jelöléseket és feltételeket felhasználva megadható ABB algoritmus paraméteres PNS feladatként történő leírása. A 3.1 ábra szemlélteti az algoritmus keretét adó *ABB* algoritmust. Az inicializálások után egy ciklus szolgáltatja a már korábban említett döntési fa bejárását. Ebben kiválasztunk egy részproblémát, amelyet megvizsgálva döntés születik arról, hogy tartalmaz-e még olyan entitást amiről nem született döntés, és szükséges-e azt

```
01. function ABB((m, o): P-graph): optimal solution
02. begin
03.  subproblems := ∅;
04.  p := P; p+ := ∅; o− := ∅; o0 := o; o+ := ∅; o* := ∅; x* := 0; z* := ∞;
05.  (m̃, õ, x̃, z̃) := SolveRelaxedLP((o−, o0, o+));
06.  if õ ≠ ∅ then
07.      subproblems := {(p, p+, o+, o−, õ, x̃, z̃)};
08.      while subproblems ≠ ∅ do
09.          (p, p+, o+, o−, õ, x̃, z̃) := SelectProblem(subproblems);
10.          subproblems := subproblems \ (p, p+, o+, o−, õ, x̃, z̃);
11.          if p = ∅ then
12.              (o*, x*, z*, subproblems) := UpdateBound(o*, x*, z*, õ, x̃, z̃,
                                                          subproblems);
13.          else
14.              subproblems := subproblems ∪ BranchABB((p, p+, o+, o−, õ, x̃, z̃));
15.          end if
16.      end while
17.  end if
18.  return(ψ(o*), o*, x*, z*);
19. end;
```

3.1. ábra. Az ABB algoritmus

tovább vizsgálni vagy sem. Ha nem, akkor átadja a részproblémát az *UpdateBound* függvénynek, amely eldönti, hogy ez az eddig megtalált legjobb megoldás-e vagy sem, és aszerint eldobja vagy eltárolja azt. Viszont ha szükséges, akkor tovább haladunk a döntési fán a korlátozó és szétválasztó feltételek segítségével, amely műveleteket a *BranchABB* függvény hajtja végre mindaddig, míg van olyan entitás, amelyről nem született valamilyen döntés, azaz, hogy bekerül-e a megoldás struktúrába vagy sem. Majd végül az egyes részproblémákat megvizsgálva megkapjuk a feladat optimális megoldását – ha létezik ilyen.

A 3.2 ábrán látható *UpdateBound* függvény tehát megvizsgálja, hogy a neki átadott, tovább már nem vizsgálható részprobléma eredménye ($\tilde{z}$) jobb-e, mint az eddig talált legjobbé (z^*). Ha igen, akkor ezután ezt tekinti az optimálisabb megoldásnak, azaz a korábbi legjobb megoldást eldobja, majd a 3.3 ábrán látható *RemoveSubproblems*

01. **function UpdateBound**($o^*, \mathbf{x}^*, z^*, \tilde{o}, \tilde{\mathbf{x}}, \tilde{z}, subproblems$)**: current best solution**
02. **begin**
03. **if** $\tilde{z} < z^*$ **then**
04. $o^* := \tilde{o}; \mathbf{x}^* := \tilde{\mathbf{x}}; z^* := \tilde{z};$
05. $subproblems := RemoveSubproblems(z^*, subproblems);$
06. **end if**
07. **return** $(o^*, \mathbf{x}^*, z^*, subproblems);$
08. **end;**

3.2. ábra. Az UpdateBound függvény algoritmusa

01. **function RemoveSubproblems**($z^*, subproblems$)**: set of subproblems**
02. **begin**
03. **for all** $(p, p^+, o^+, o^-, \tilde{o}, \tilde{\mathbf{x}}, \tilde{z}) \in subproblems$ **where** $\tilde{z} > z^*$ **do**
04. $subproblems := subproblems \setminus (p, p^+, o^+, o^-, \tilde{o}, \tilde{\mathbf{x}}, \tilde{z});$
05. **end for**
06. **return** $subproblems;$
07. **end;**

3.3. ábra. A RemoveSubproblems függvény

függvénnyel kiszűri a még vizsgálatra váró részproblémák közül azt, amelyik már biztosan nem adhat jobb eredményt, mint a már korábban megtalált, és eldobja azokat, ezzel is tovább szűkítve a keresési teret. Viszont ha nem jobb, akkor az átvett részproblémát dobja el.

A 3.4 ábrán látható *SelectProblemABB* függvényt a 3.1 ábrán tálalható ABB algoritmus használja a soron következő legalkalmasabb részprobléma kiválasztásra. Mindig azt a részproblémát választja a *subproblems* halmazból amelyiknek a legkisebb az aktuális optimum értéke.

A 3.5 ábrán látható *BranchABB* függvény végzi a döntési fa bejárását a következőképpen. A még nem vizsgált p entitások közül kiválaszt egyet, az x-et, majd a segítségével összegyűjti az összes olyan tevékenységet, amely eredményezni tudja azt,

```
01. function SelectProblemABB(𝒮: set of subproblems): subproblem
02. begin
03.  return (p, p⁺, o⁺, o⁻, õ, x̃, z̃) ∈ 𝒮 where z̃ is minimal;
04. end;
```

3.4. ábra. A SelectProblemABB függvény algoritmusa

és még nem lett se kizárva se bevonva a struktúrába. Ezután generálja ezeknek a tevékenységeknek a $\mathcal{C}$ hatványhalmazát, majd a hatványhalmaz minden elemére elvégzi a korlátozó és szétválasztó műveleteket, amelyek között szerepel gyorsításként egy neutrális kiterjesztés is. Amennyiben a beválasztott és kizárt tevékenységeknek nincs közös metszete, akkor megoldjuk az aktualizált részproblémára a relaxált LP feladatot a *SolveRelaxedLP* függvénnyel, majd ha a kapott eredmény tevékenység halmaza nem üres, akkor aktualizáljuk a részproblémák halmazát, egyébként eldobja a részproblémát és nem vizsgálja tovább.

A 3.6 ábrán látható NX algoritmus a neutrális kiterjesztést valósítja meg, amely egy gyorsítása az ABB algoritmusnak. Ez a gyorsítás úgy működik, hogy ha egy struktúrába bevett tevékenység valamely bemenetét korábbi döntésekkel összhangban legfeljebb egy tevékenység biztosíthatja, akkor azt is bevesszük a struktúrába.

A 3.7 ábrán látható RSG algoritmus szintén egy gyorsírása az ABB algoritmusnak. Az RSG minden olyan tevékenységet kizár, melynek valamely bemenete korábbi döntések alapján nem előállítható. Ha van ilyen korábban bevett tevékenység, akkor a korábbi döntések ellentmondáshoz vezetnek. Ez esetben a részprobléma eldobható.

A 3.8 ábrán található *SolveRelaxedLP* függvény végzi a relaxált LP feladat megoldását, a korábbiakban bemutatott feltételek felhasználásával. Ez egy minimalizálási LP feladat, amely abban az esetben ha megoldható, akkor a struktúra aktuálisan kiszámolt értékével tér vissza, ha pedig nem, akkor üres entitás és tevékenység halmazokkal, amelyek hatására a *BranchABB* függvény eldobja a struktúrát és tovább nem vizsgálja azt.

```
01. function BranchABB((p, p+, o+, o−, õ, x̃, z̃):subproblem):set of
                    subproblems
02. begin
03.  subp := ∅;
04.  Let x ∈ p;
05.  o_x := φ−({x}) \ o−; o_xb:= (o_x ∩o+);
06.  C := ℘(o_x \ o_xb);
07.  if o_xb := ∅ then
08.       C := C \ {∅};
09.  end if
10.  for all c ∈ C do
11.       p̂ := p ∪ ψ−(c) \ p+ \  {x} \ R;
12.       p̂+ := p+ ∪ {x};
13.       (m, o):= RSG(o− ∪ (O_x \ O_xb \ c));
14.       ô− := O \ o;
15.       ô+:= NX(o+ ∪ c);
16.       if (o+ ∩ o− = ∅) then
17.            Δp := p̂ ∩ ψ+(ô+) \ ψ+(O \ ô− \ ô+);
18.            p̂ := p̂ \ Δ(p);
19.            p̂+ := p̂+ ∪ Δ(p);
20.            (m̃, õ, x̃, z̃):= SolveRelaxedLP(ô−, ô+, o \ ô− \ ô+);
21.            if (õ ≠ ∅) then
22.                 subp := subp∪ (p̂, p̂+, ô+, ô−, õ, x̃, z̃);
23.            end if
24.       end if
25.  end for
26.  return subp;
27. end;
```

3.5. ábra. A BranchABB függvény algoritmusa

```
01. function NX(o+, o0:sets of operating units):set of operating units
02. begin
03.  repeat
04.        Δo := ∅;
05.        m = ψ−(o+) \ ψ+(o+) \ R;
06.        for all mi ∈ m where |φ−({mi}) ∩ o0| = 1 do
07.              Δo := Δo ∪ (φ−({mi}) ∩ o0);
08.        end for
09.        o+ := o+ ∪ Δo;
10.  until Δo = ∅
11.  return o+;
12. end;
```

3.6. ábra. Az NX algoritmus

```
01. function RSG(o−:set of operating units):P-graph
02. begin
03.  o := O \ φ−(R) \ o−;
04.  r := ψ−(o) \ ψ+(o) \ R;
05.  comment: reduction part
06.  while r ≠ ∅ do
07.        o := o \ φ+(r);
08.        r := ψ−(o) \ ψ+(o) \ R;
09.  end while
10.  if P ⊄ ψ+(o) then
11.        return;
12.  end if
13.  comment: construction part
14.  p := P; o' := ∅; Δo := ∅;
15.  repeat
16.        Δo := φ−(p) ∩ o \ o';
17.        o' := o' ∪ Δo;
18.        p := ψ−(Δo);
19.  until Δo ≠ ∅
20.  return(o', ψ(o'));
21. end;
```

3.7. ábra. Az RSG algoritmus

01. **function SolveRelaxedLP($\mathsf{o}^-, \mathsf{o}^0, \mathsf{o}^+$:sets of operating units):solution of the relaxed problem**
02. **begin**
03. $\tilde{\mathbf{L}} := \mathbf{L}$; $\tilde{\mathbf{U}} := \mathbf{U}$;
04. **for all** $(\alpha, \beta) = o_i \in \mathsf{O}$
05. **begin**
06. $\tilde{l_i} := \begin{cases} 0 & \text{if } o_i \in \mathsf{o}^- \cup \mathsf{o}^0 \\ \varepsilon > 0 & \text{if } o_i \in \mathsf{o}^+ \end{cases}$
07. $\tilde{u_i} := \begin{cases} u_i & \text{if } o_i \in \mathsf{o}^0 \cup \mathsf{o}^+ \\ 0 & \text{if } o_i \in \mathsf{o}^- \end{cases}$
08. $\tilde{c_i} := \begin{cases} cp_i - \sum_{m_j \in \alpha \cup \beta} a_{ji} cm_j & \text{if } o_i \in \mathsf{o}^+ \\ cp_i - \sum_{m_j \in \alpha \cup \beta} a_{ji} cm_j + \frac{cf_i}{u_i} & \text{if } o_i \in \mathsf{o}^0 \end{cases}$
09. **end**
10. **Linear Program:**
$\tilde{\mathbf{z}} = \mathbf{c}^{\mathbf{T}} \tilde{\mathbf{x}} \to \min$
$\tilde{\mathbf{L}} \leq \mathbf{A}\tilde{\mathbf{x}} \leq \tilde{\mathbf{U}}$
11. $\tilde{\mathbf{l}} \leq \tilde{\mathbf{x}} \leq \tilde{\mathbf{u}}$
12. **end Linear Program**
13. **if** Linear Program is feasible **then**
14. Solve Linear Program;
15. $\tilde{\mathsf{o}} := \{o_i \in \mathsf{O} : \tilde{x_i} > 0\}; \tilde{z} := \tilde{z} + \sum_{o_i \in \mathsf{o}^+} cf_i;$
16. **else**
17. $\tilde{o} := \emptyset;$
18. **end if**
19. **return** $(\psi(\tilde{\mathsf{o}}), \tilde{\mathsf{o}}, \tilde{\mathbf{x}}, \tilde{z});$
20. **end;**

3.8. ábra. Az SolveRelaxedLP függvény algoritmusa

4. fejezet

A PNS egymást kölcsönösen kizáró tevékenységek kezelésére

A most következő fejezetben a PNS módszertan kölcsönösen kizáró tevékenységek kezelésére kidolgozott kiegészítése kerül bemutatásra.

4.1. Feladat leírás

Sok esetben nem elegendő csak az optimális struktúra felépítését szem előtt tartanunk, hiszen számos egyéb körülmény is hatással lehet folyamatainkra. Egy nagyon egyszerű esetet említve optimális struktúra részeként kaphatjuk például a következő erőentitásokat:

- A rendszerünk operációs rendszere legyen valamelyik Linux disztribúció.
- A rajta futó levelező szoftver pedig MS Exchange legyen.

Látjuk, hogy hiába kaptuk meg az optimális megoldást, abban találhatóak olyan előírások, melyeket nem tudunk biztosítani (nem adható meg Linux rendszerre MS Exchange levelező program). Az ilyen és ehhez hasonló esetek kezelésére szükséges a P-gráf módszertan egy módosított változatának bevezetése.

Definíció 4.1.1 *Legyen O a tevékenységek véges és nem üres halmaza valamint $OX \subseteq \wp(O)$, ahol $\wp(O)$ jelöli O hatványhalmazát és $\forall ox \in OX$ esetén teljesül, hogy $|ox| > 1$.*

Az $\mathcal{OX}$ a tevékenységek olyan halmazainak halmazát mutatja, amelyek nem szerepelnek egyazon struktúrában.

Az eredeti PNS feladat öt kombinatorikus axiómáját, amelyeket a 2.2.3 fejezetben mutattam be, kiegészítettem egy folyamatszintézis során vizsgálandó újabb strukturális feltétellel ami garantálja, hogy kölcsönösen kizáró tevékenységek nem kerülhetnek bele egyetlen egy lehetséges megoldás struktúrába sem ([104], [62]). A PNS feladat definíciója a következőképpen módosul.

Definíció 4.1.2 *Legyen adott egy PNS-feladat $(\mathcal{P}, \mathcal{R}, \mathcal{O}, \mathcal{OX})$ strukturális modellje és legyen (m, o) P-gráf az $(\mathcal{M}, \mathcal{O})$ P-gráf részgráfja. Az (m, o) részgráfot kölcsönös kizárási feltételeket is teljesítő kombinatorikusan lehetséges struktúrának nevezzük, amely mentes a kölcsönösen kizáró tevékenységektől, ha teljesül az S1 - S5 feltételek mindegyike továbbá a megkövetelt új strukturális feltétel, ahol:*

- *Egyetlen egy megoldás struktúra sem tartalmaz kölcsönösen kizáró tevékenységeket, azaz $\forall ox \in \mathcal{OX} : |o \cap ox| \leq 1$.*

A új feltétel bevezetésével tovább csökkenhet a kombinatorikusan lehetséges struktúrák száma. Ezért ezen új feltétel felhasználásával az SSG algoritmust továbbfejlesztettem, amelyet az SSGMX (Solution Structure Generator with Mutual Exclusions) algoritmusként mutatok be a következőkben. Az új strukturális feltétel az optimális megoldások precízebb felderítésében is nagy segítséget nyújt, így ezt a tulajdonságot integráltam az ABB algoritmusba is, amelyet ABBMX (Accelerated Branch-and-Bound with Mutual Exclusions) algoritmusként szemléltetek.

4.2. Az SSGMX algoritmus

Az eredeti PNS algoritmusok nem foglalkoznak a kölcsönösen kizáró tevékenységekkel, és mivel a komplex folyamatok esetében szükséges lehet ezeknek a megfelelő kezelése ezért szükséges volt a módszertan kiegészítése. Ebben az alfejezetben a kölcsönösen kizáró tevékenységek kezelésével kiegészített SSG algoritmus kerül bemutatásra, amely az SSGMX algoritmus nevet kapta és a 4.1 és 4.2 ábrákon látható.

```
01. function SSGMX((m, o): P-graph): set of solution structures
02. begin
03.  solutions := ∅;
04.  subproblems := {(P, ∅, ∅, ∅)};
05.  while subproblems ≠ ∅ do
06.      let (p, p+, o+, o−) ∈ subproblems;
07.      subproblems := subproblems \ (p, p+, o+, o−);
08.      if p = ∅ then
09.          solutions := solutions ∪ {(ψ(o+), o+)};
10.      else
11.          BranchSSGMX((p, p+, o+, o−), subproblems);
12.      end if
13.  end while
14.  return solutions;
15. end;
```

4.1. ábra. Az SSGMX függvény algoritmusa

Az SSGMX algoritmus is egy rekurzív magra épül, ahogy azt már az eljárás alapjául szolgáló SSG algoritmus esetében is tapasztalhattuk a 2.2.5 fejezetben. Látható, hogy ez az algoritmus is alulról építkezik. Az összes kombinatorikusan lehetséges megoldásstruktúrát egy ciklus segítségével állítja elő, amely ciklust az SSGMX függvény tartalmazza, ahogy az a 4.1 ábrán látható. Ez a függvény adja az eljárás keretét, amely a BranchSSGMX függvényben (4.2 ábra) található különböző döntési leképezések hatására a cikluson belül egy rekurzív függvényhívást használ. Ez mindig végcéloktól indul, lépésenként az őket eredményező tevékenységeket beválasztva, amíg el nem jut az erőforrásokig. A BranchSSGMX a 3. lépésben különbözik az eredeti BranchSSG függvénytől. Ez a lépés az a kiegészítés, ami az egymást kizáró tevékenységek kérdését kezeli a következőképpen. Az eljárás megvizsgálja, hogy az $\mathcal{OX}$ halmazban található kizáró $o\mathcal{X}$ halmazok tevékenységeiből választottunk-e be legalább egyet az aktuális struktúrába. Ha igen, és kettő vagy annál több tevékenységet, akkor az a megoldás hamis és eldobjuk. Ez úgy történik, hogy a fizibilitást jelző *feasible* változót *false* értékre állítjuk, amelyet majd a 4. lépésben felhasználva vagy eldobjuk, vagy a segítségével tovább építjük a struktúrát. Ha pedig csak egy tevékenység

```
01.      function BranchSSGMX((p, p+, o+, o−): subproblem,
                                   subproblems: set of subproblems)
02.      begin
03. 1. lépés  Let x ∈ p;
04.           o_x := φ−({x}) \o−;
05.           o_xb := (o_x ∩o+);
06.           C := ℘(o_x \ o_xb);
07.           if o_xb := ∅ then
08.                C := C \ {∅};
09.           end if
10. 2. lépés  for all c ∈ C do
11.                ô+ := o+ ∪ {c};
12.                ô− := o−;
13.                feasible := true;
14. 3. lépés       for all oχ ∈ OX do
15.                     if oχ ∩ ô+ ≠ ∅ then
16.                          if |oχ ∩ ô+| > 1 then
17.                               feasible := false; comment: infeasible solution
18.                               break;
19.                          else
20.                               ô− := ô− ∪ oχ \ (oχ ∩ ô+);
21.                          end if
22.                     end if
23.                end for
24. 4. lépés       if feasible then
25.                     subproblems := subproblems ∪ ((p ∪ ψ−(c)) \ p+ \ {x} \ R,
                                 p+ ∪ {x} , ô+ , ô− ∪ (o_x \ o_xb \ c) ));
26.           end for
27.      end;
```

4.2. ábra. A BranchSSGMX függvény algoritmusa

került be korábban az épülő struktúrába, akkor az azt tartalmazó kizáró halmazok összes többi tevékenységét hozzáadjuk a kizárt tevékenységek $\hat{o}^-$ halmazához. Az SSGMX előállít tehát minden egyes olyan kombinatorikusan lehetséges esetet, amely nem tartalmaz kölcsönösen kizáró tevékenységeket.

4.2.1. Az SSGMX algoritmus helyességének bizonyítása

Az algoritmus helyességének bizonyításához felhasználom az alapjául szolgáló SSG algoritmus helyesség bizonyítását, amely alapján megfogalmazható a 4.2.1 tétel. Majd ezután a 4.2.2 tételben belátom, hogy az SSGMX algoritmus az SSG algoritmus kölcsönösen kizáró tevékenységek kezelésével kibővített változata.

Tétel 4.2.1 *Az SSGMX algoritmus az alapjául szolgáló SSG algoritmus összes tulajdonságával rendelkezik, azaz minden kombinatorikusan lehetséges struktúrát generál, az általa generált struktúrák kombinatorikusan lehetségesek, továbbá minden kombinatorikusan lehetséges struktúrát pontosan egyszer generál és véges futású.*

Bizonyítás *A 4.2.1 tétel bizonyítása egyértelműen következik a SSG algoritmus bizonyításából.*

Tétel 4.2.2 *Az SSGMX algoritmus minden olyan kombinatorikusan lehetséges struktúrát, amely nem tartalmaz kölcsönösen kizáró tevékenységeket, pontosan egyszer generál. Az eredményül kapott struktúrák mind kombinatorikusan lehetségesek és nem tartalmaznak kölcsönösen kizáró tevékenységeket.*

Bizonyítás *Az SSGMX algoritmus a kölcsönösen kizáró tevékenységek kezelésével lett kiegészítve az alapjául szolgáló SSG algoritmushoz képest. Ez a kiegészítés egyrészt megjelenik a modellben ahol a $\mathcal{P}$ (végcélok) és $\mathcal{R}$ (erőforrások) halmazokon felül az $\mathcal{OX}$ (kölcsönösen kizáró tevékenységek halmazai) halmaz is adott. Másrészt pedig helye van az adott részprobléma fizibilitás vizsgálatakor, ami nem más, mint az újonnan bevezetett kiegészítő strukturális feltétel integrálása az alap algoritmusba. Ezt a kiegészítést a 4.2 ábrán látható BranchSSGMX függvény 3. lépése azonosítja, amely a következőkből áll. Az eljárás megvizsgálja, hogy valamely kölcsönösen kizáró*

halmazból került-e már tevékenység az aktuálisan vizsgált részprobléma struktúrájába. Abban az esetben ha még nem került egyetlen egy olyan tevékenység sem a struktúrába, amelyiknek definiált legalább egy kölcsönösen kizáró társa, akkor nincs plusz teendő, halad tovább az eljárás. Viszont, ha azonosítható legalább egy ilyen tevékenység, akkor az további vizsgálatot igényel, amely vizsgálat során két eset állhat elő. Az egyik az, amikor egynél több kölcsönösen kizáró tevékenység került az adott részprobléma struktúrájába. Ez egy olyan állapotot eredményez, amely nem vezethet megengedett megoldáshoz. Ebben az esetben az SSGMX algoritmus nem folytatja a részprobléma további vizsgálatát, azt eldobja. A másik eset az, amikor bármely kölcsönösen kizáró halmazból maximum egy tevékenységet tartalmaz az adott részprobléma struktúrája, és amelyik kölcsönösen kizáró ox halmazból tartalmaz tevékenységet, annak az összes többi elemét beleteszi a kizárt tevékenységek $\hat{o}^-$ *halmazába, hogy ne kerülhessenek bele a struktúrába és ezzel ne okozhassanak ellentmondást. Tehát az SSGMX algoritmus az SSG algoritmus által generált kombinatorikusan lehetséges struktúrák közül csak azokat a struktúrákat zárja ki, amelyek a feladat megfogalmazásában szereplő szükséges feltételek alapján soha nem valósulhatnának meg.*

4.3. A MILP modell

Mint azt korábban említettük, az ABB algoritmus döntési leképezéseken alapul. Azonban a döntés meghozatala előtt elvégez egy korlátszámítási eljárást, mely egy alsó korlátot rendel minden részproblémához, és azok ezen érték alapján elvethetőek, vagy megtarthatóak a megoldás további keresésének szempontjából. Ez a korlátszámítás egy matematikai programozási modell megoldását foglalja magába. A kölcsönös kizárás hatékony működése, valamint a korlátok további validitása miatt képessé kell tenni ezt a MILP modellt is a kölcsönösen kizáró halmazokkal történő számolásra. A modell megoldási módjának részletes bemutatásával a következő fejezetben foglalkozunk.

Legyen y_i egy bináris változó, amelynek az értéke 1, ha az o_i tevékenység szerepel az adott megoldás struktúrában, és 0 az értéke ha nem. Mivel szerepelhetnek kölcsönösen kizáró halmazok a feladatban, amelyek mindegyikéből csak egyetlen elem

```
01. function ABBMX((m, o): P-graph): optimal solution
02. begin
03.  subproblems := ∅;
04.  p := P; p+ := ∅; o− := ∅; o0 := o; o+ := ∅; o* := ∅; x* := 0; z* := ∞;
05.  (m̃, õ, x̃, z̃) := SolveRelaxedLP((o−, o0, o+));
06.  if õ ≠ ∅ then
07.      subproblems := {(p, p+, o+, o−, õ, x̃, z̃)};
08.      while subproblems ≠ ∅ do
09.          (p, p+, o+, o−, õ, x̃, z̃) := SelectProblem(subproblems);
10.          subproblems := subproblems \ (p, p+, o+, o−, õ, x̃, z̃);
11.          if p = ∅ then
12.              (o*, x*, z*, subproblems) := UpdateBound(o*, x*, z*, õ, x̃, z̃,
                                                 subproblems);
13.          else
14.              subproblems := subproblems ∪ BranchABBMX((p, p+, o+,
                                                 o−, õ, x̃, z̃));
15.          end if
16.      end while
17.  endif
18.  return(ψ(o*), o*, x*, z*);
19. end;
```

4.3. ábra. Az ABBMX függvény algoritmusa

szerepelhet egyszerre a megoldásstruktúrában, emiatt egy új felső korlát alkalmazható ezen halmazokban levő y_i összegeire. Azaz teljesülnie kell minden ilyen esetben az alábbi egyenlőségnek:

$$\forall o\mathcal{X} \in \mathcal{OX} : \sum_{o_i \in o\mathcal{X}} y_i = \sum_{o_i \in o\mathcal{X}} \frac{x_i}{\widetilde{u}_i} \leq 1 \tag{4.3.1}$$

4.4. Az ABBMX algoritmus

A 4.3 ábrán látható ABBMX algoritmus a 3.4 fejezetben bemutatott ABB algoritmuson alapul, amely a kölcsönösen kizáró műveletek kezelésével lett kiegészítve. Ehhez a *BranchABB* és a *SolveRelaxedLP* függvényeket kellett kiegészíteni, illetve egy

új MX függvényt vezettem be. A 4.4 ábrán látható a $BranchABBMX$ függvény, amely a $BranchABB$ függvény kölcsönösen kizáró műveleti egységek kezelésével kibővített változata. A legszignifikánsabb különbség a 3. lépés, amely egy hátul tesztelő ciklust azonosít. Ebbe a ciklusba került bele a már korábban ismertetett RSG és NX függvények mellett az újonnan bevezetett MX függvény meghívása is. A ciklus addig működik, amíg sikerül újabb tevékenységről döntést hoznia aszerint, hogy a tevékenységet kizárja vagy beveszi a struktúrába, továbbá a kizárt és a bevont tevékenységek halmazainak nincs közös elemük. Miután már nem tud több tevékenységről döntést hozni, a 4. lépésben ha nincs közös eleme a kizárt és a bevont tevékenységek halmazainak, akkor megpróbálja megoldani a módosított relaxált LP feladatot a $SolveRelaxedLPMX$ függvénnyel, amely a $SolveRelaxedLP$ függvény kölcsönösen kizáró műveleti egységek kezelésével kibővített változata. Ha sikerül megoldani, akkor az új részproblémát eltároljuk a részproblémák halmazában. Miután a függvény a $\mathcal{C}$ hatványhalmaz minden c elemére elvégzi a struktúra vizsgálatát, és ha lehetséges a bővítését, visszaadja a frissített részproblémák halmazát.

A 4.5 ábrán látható $SolveRelaxedLPMX$ függvény a $SolveRelaxedLP$ fügvény 4.3.1 egyenlettel kibővített változata. Azaz a korlátozó feltételek közé bekerült egy felső korlát a kölcsönösen kizáró tevékenységekből beválasztható tevékenységek számosságára. Ez a felső korlát az 1, azaz maximum 1 tevékenység szerepelhet a struktúrában a kölcsönösen kizáró műveletek közül.

A 4.6 ábrán szemléltetett MX függvény végzi a részproblémák kölcsönösen kizáró műveletek szerinti vizsgálatát. A függvény megvizsgálja, hogy az összes $\mathcal{OX}$ halmazbeli kölcsönösen kizáró $o\chi$ tevékenységhalmaz valamelyikéből szerepel-e az $\hat{o}^+$ beválasztott tevékenységek között legalább egy. Abban az esetben ha igen, és több mint egy tevékenység szerepel a beválasztott tevékenységek között valamelyik $o\chi$ kölcsönösen kizáró halmazból, akkor az hibás megoldás és a beválasztott tevékenységek halmazát adja vissza a függvény, amely azt eredményezi, hogy a részprobléma nem lesz tovább vizsgálva. Viszont ha csak egy tevékenység szerepel a beválasztott tevékenységek között valamelyik $o\chi$ halmazból, akkor azon egy tevékenység kivételével az adott $o\chi$ halmaz összes többi eleme bekerül a kizárt tevékenységek halmazába.

```
01.        function BranchABBMX((p, p+, o+, o−, õ, x̃, z̃ ):subproblem): set of
02.                                   subproblems
03.        begin
04. 1. lépés   subp := ∅;
05.            Let x ∈ p;
06.            o_x := φ−({x}) \ o−; o_xb := (o_x ∩ o+);
07.            C := ℘(o_x \ o_xb);
08.            if o_xb := ∅ then
09.                C := C \ {∅};
10.            end if
11. 2. lépés   for all c ∈ C do
12.                p̂ := p ∪ ψ−(c) \ p+ \ {x} \ R;
13.                p̂+ := p+ ∪ {x};
14.                ô2− := o−;
15.                ô2+ := o+ ∪ c;
16. 3. lépés       do
17.                    ô− := ô2−;
18.                    ô+ := ô2+;
19.                    (m, o) := RSG(ô− ∪ (O_x \ O_xb \ c));
21.                    ô2− := O \ o;
22.                    ô2+ := NX(ô+);
23.                    ô2− := MX(ô2−, ô2+);
24.                while ((ô2− ≠ ô− or ô2+ ≠ ô+) and ô2+ ∩ ô2− = ∅);
25. 4. lépés       if (ô2+ ∩ ô2− ≠ ∅) then
26.                    Δp := p̂ ∩ ψ+(ô+) \ ψ+(O \ ô− \ ô+);
27.                    p̂ := p̂ \ Δ(p);
28.                    p̂+ := p̂+ ∪ Δ(p);
29.                    (m̃, õ, x̃, z̃) := SolveRelaxedLPMX(ô−, ô+, o \ ô− \ ô+);
30.                    if(õ ≠ ∅) then
31.                        subp := subp ∪ (p̂, p̂+, ô+, ô−, õ, x̃, z̃);
32.                    end if
33.                end if
34.            end for
35.            return subp;
36.        end;
```

4.4. ábra. A BranchABBMX függvény algoritmusa

01. **function SolveRelaxedLPMX**(o^-, o^0, o^+**:sets of operating units):**
02. **solution of the relaxed problem**
03. **begin**
04. $\tilde{\mathbf{L}} := \mathbf{L}$; $\tilde{\mathbf{U}} := \mathbf{U}$;
06. **for all** $(\alpha, \beta) = o_i \in O$
07. **begin**
08. $\tilde{l_i} := \begin{cases} 0 & \text{if } o_i \in o^- \cup o^0 \\ \varepsilon > 0 & \text{if } o_i \in o^+ \end{cases}$
09. $\tilde{u_i} := \begin{cases} u_i & \text{if } o_i \in o^0 \cup o^+ \\ 0 & \text{if } o_i \in o^- \end{cases}$
10. $\tilde{c_i} := \begin{cases} cp_i - \sum_{m_j \in \alpha \cup \beta} a_{ji} cm_j & \text{if } o_i \in o^+ \\ cp_i - \sum_{m_j \in \alpha \cup \beta} a_{ji} cm_j + \frac{cf_i}{u_i} & \text{if } o_i \in o^0 \end{cases}$
11.
12. $\forall o\mathcal{X} \in O\mathcal{X}$: $\sum_{o_i \in o\mathcal{X}} \frac{x_i}{\tilde{u}_i} \leq 1$
13. **end**
14. **Linear Program:**
15. $\tilde{\mathbf{z}} = \mathbf{c}^{\mathbf{T}} \tilde{\mathbf{x}} \rightarrow \min$
16. $\tilde{\mathbf{L}} \leq \mathbf{A}\tilde{\mathbf{x}} \leq \tilde{\mathbf{U}}$
17. $\tilde{\mathbf{l}} \leq \tilde{\mathbf{x}} \leq \tilde{\mathbf{u}}$
18. **end Linear Program**
19. **if** Linear Program is feasible **then**
20. Solve Linear Program;
21. $\tilde{o} := \{o_i \in O : \tilde{x_i} > 0\}; \tilde{z} := \tilde{z} + \sum_{o_i \in o^+} cf_i;$
22. **else**
23. $\tilde{o} := \emptyset;$
24. **end if**
25. **return** $(\psi(\tilde{o}), \tilde{o}, \tilde{\mathbf{x}}, \tilde{z});$
26. **end;**

4.5. ábra. A SolveRelaxedLPMX függvény algoritmusa

```
01. function MX(ô⁻, ô⁺: sets of operating units): set of operating units
02. begin
03.  for all oχ ∈ OX do
04.      if oχ ∩ ô⁺ ≠ ∅ then
05.          if |oχ ∩ ô⁺| > 1 then
06.              return ô⁺; comment: infeasible solution
07.          else
08.              ô⁻ := ô⁻ ∪ oχ \ (oχ ∩ ô⁺);
09.          end if
10.      end if
11.  end for
12.  return ô⁻;
13. end;
```

4.6. ábra. Az MX függvény algoritmusa

Majd végül, ha nem talált olyan $\mathcal{OX}$ halmazbeli kölcsönösen kizáró $o\chi$ tevékenységhalmazt, amelyből egynél több tevékenység szerepelne az adott részprobléma beválasztott tevékenységei között, akkor az visszaadja a kizárt tevékenységek halmazát a $BranchABBMX$ függvénynek.

4.4.1. Az ABBMX algoritmus helyességének bizonyítása

Az algoritmus helyességének bizonyításához felhasználom az alapjául szolgáló ABB algoritmus illetve az SSGMX algoritmus helyesség bizonyítását, amely során belátom, hogy az ABBMX algoritmus az ABB algoritmus kölcsönösen kizáró tevékenységek kezelésével kibővített változata.

Tétel 4.4.1 *Az ABBMX algoritmus futási eredményeként megadja azt a valamely szempont szerinti (költség, stb.) optimális struktúrát, amely nem tartalmaz kölcsönösen kizáró tevékenységeket, és azt pontosan egyszer generálja.*

Bizonyítás *Az ABBMX algoritmus a kölcsönösen kizáró tevékenységek kezelésével lett kiegészítve az alapjául szolgáló ABB algoritmushoz képest. Az ABBMX algoritmus ugyanúgy az optimális megoldást találja meg, mint az ABB algoritmus, viszont ha*

a feladat tartalmaz kölcsönösen kizáró tevékenységeket, akkor az újonnan bevezetett kiegészítés lényegesen leszűkítheti a keresési teret, és így az algoritmus futási idejét. Ez a kiegészítés egyrészt ugyanúgy megjelenik a PNS modellben, mint az SSGMX esetében, azaz a $\mathcal{P}$ (végcélok halmaza) és $\mathcal{R}$ (erőforrások halmaza) halmazokon felül az $\mathcal{OX}$ kölcsönösen kizáró tevékenységek halmazainak halmaza is adott. Másrészt pedig az adott részprobléma fizibilitás vizsgálatakor, ami nem más, mint az újonnan bevezetett kiegészítő strukturális feltétel integrálása az alap algoritmusba. Ezt a kiegészítést a 4.6 ábrán látható MX függvény azonosítja, amely ugyanolyan módon működik ahogy azt már a 4.2.1 alfejezetben az SSGMX algoritmus helyesség bizonyítása során a kölcsönös kizárás kezelésére bemutattam.

Tétel 4.4.2 *Abban az esetben, ha az optimalizálandó feladat tartalmaz kölcsönösen kizáró tevékenységeket az ABBMX algoritmus az ABB algoritmusnál kevesebb lépésben, azaz gyorsabban találja meg a feladat optimális megoldását, ha ilyen létezik.*

Bizonyítás *Az ABBMX algoritmusnak a kölcsönös kizárásokat figyelembe véve kevesebb részproblémát kell ahhoz megvizsgálni, hogy megtalálja a feladat optimális megoldását, mivel a kölcsönös kizárási feltételek kezelésének köszönhetően, eleve kizár olyan részstruktúrákat a keresési térből, amelyeket az eredeti ABB algoritmus nem tud megtenni. Az ABB algoritmus esetén, csak miután megkaptuk az eredményt, egy utólagos szűrés után derül ki, hogy a kapott eredmény kielégíti-e a kölcsönös kizárási feltételeket vagy sem. Ha nem, akkor a további vizsgálatok szükségesek, míg megtaláljuk azt a legoptimálisabb megoldást, ami már a kölcsönös kizárási feltételeket is kielégíti. Végül mind a két esetben ugyanarra a megoldásra jutunk, viszont látható, hogy mennyivel hosszadalmasabb és körülményesebb az ABB algoritmus használata az ABBMX algoritmussal szemben abban az esetben, ha kölcsönös kizárási feltételeket is figyelembe kell venni.*

4.5. A kölcsönös kizárás kezelését szemléltető jármű hozzárendelési példa

Az előzőekben bemutatott kölcsönös kizárás által nyújtott keresési tér csökkenés kihasználhatóságát egy jármű hozzárendelési problémán keresztül fogom reprezentálni. A jármű hozzárendelési probléma (Vehicle Assignemnt Problem, VAP) egy specializált hozzárendelési probléma, melyben különböző elvégzendő szállítási feladatok végrehajtása céljából rendelünk hozzájuk járműveket. A problémában az elvégzendő szállítások, a rendelkezésre álló járművek, valamint a környezet (távolságok, helyek, stb.), és minden egyéb, a feladat során szükséges tulajdonság (jármű kapacitás, maximális sebesség, fogyasztás, stb.) előre definiált, továbbá az is, hogy mely jármű mely feladatok elvégzésére képes. A feladat egy olyan hozzárendelés megvalósítása, melyben az összes szükséges feladatot elvégezzük, és az valamely előre meghatározott szempontból optimális. A feladatnak többféle megoldási módja is megtalálható a szakirodalomban, a heurisztikus módszereken keresztül a bizonyítottan optimális eredményt szolgáltatóig. Röviden bemutatásra kerül egy olyan bizonyítottan optimális megoldási módszer, mely PNS problémaként kezeli a VAP problémát, és így képes a hatékony megoldás meghatározására ([6],[7]).

A modellkonverzió alapelgondolása a következő. Tekintsünk az elvégzendő szállítási feladatokra úgy, mint végcélokra. A erőforrások pedig legyenek a kiindulási helyükön állomásozó használható járművek. A köztes entitások jelöljék a szállítások során érinthető helyeket, esetleg átpakolási, vagy pihenőhelyeket, amiket érintenie kell, vagy érinthet az adott jármű minden esetben. A tevékenységek ezen analógia alapján az egyes helyek közti lehetséges mozgásokat fogják reprezentálni. A tevékenységek magukba foglalják az egyes járművek használatba hozatalát, a járművek különböző helyek közti mozgatását és a járművek célba juttatását egy olyan helyről, amelyből vezet út a kívánt célhoz. A modellkonverzió pontos leírása megtalálható ([6]). Ezek alapján egy VAP probléma PNS problémaként kezelhető.

Ahogy azt előzőekben bemutattuk a VAP problémának előállítható a PNS modellje, amely a P-gráf keretrendszer segítségével, egyszerűen kezelhető, valamint a rendelkezésre álló algoritmusokkal meghatározható az optimális megoldása, és akár

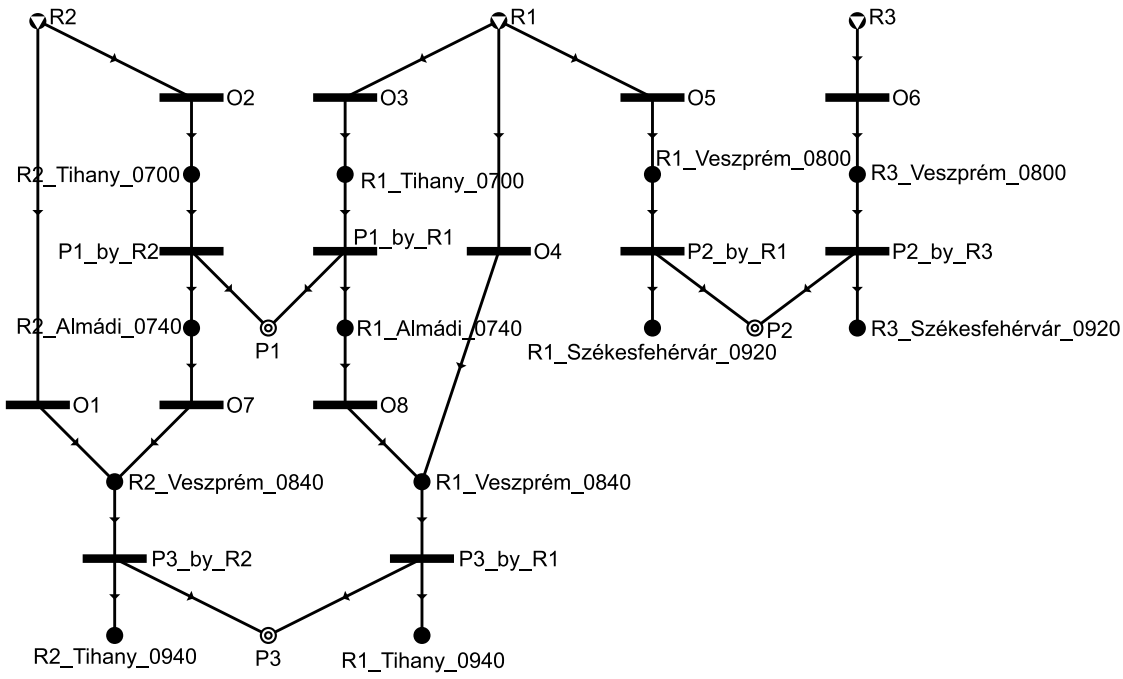
az n-legjobb megoldása is valamely szempontból. Azonban ez a fajta megoldási mód figyelmen kívül hagyja a kölcsönösen kizáró halmazokat, és nem veszi őket figyelembe a modell generálása során sem. A kölcsönösen kizáró halmazok modellbe foglalása nem befolyásolja a kapott eredményeket optimalitás szempontjából, csupán a megoldási időt, valamint a megoldás során generált részproblémák számát képes jelentős mértékben csökkenteni. Ez az újonnan bevezetett P-gráf módszertanbeli kiegészítéseknek köszönhető, mely képes figyelembe venni a kölcsönös kizárást.

A továbbiakban bemutatásra kerül, hogy hol találkozhatunk kölcsönösen kizáró tevékenység halmazokkal egy VAP-PNS modellben. Elsőként logikai megfontolásként megemlítendő, hogy a hozzárendelések során egy időben egy jármű nem lehet egyszerre több helyen (nem hajthat végre egyszerre utazást két közbülső hely között ugyanabban az időintervallumban), ez fizikai képtelenség lenne. Ha egy jármű képes a kiinduló telephelyéről több köztes célállomásba szállítani, és ezek közt találhatók olyanok, melyek szállítási időintervallum metszete létezik, akkor ezekre a tevékenységekre tekinthetünk, mint kölcsönösen kizáró elemekre.

Továbbá az egyes tevékenységek kölcsönösen kizáróak, ha közös kiindulási pontból szállít ugyanaz a jármű, és a szállítások időintervallumának van metszete. Emellett még abban az esetben is, ha különböző helyekről szállítana ugyanaz a jármű egy közös célba, valamint a legszélsőségesebb eset, amikor ugyanaz a jármű ugyanazon időben különböző kiindulási pontból szállítana különböző célokba. Azon tevékenységek halmaza, melyek valamely esetet megvalósítják, azoknak kölcsönösen kizárónak kell lennie.

A célba szállítást reprezentáló tevékenységek, akkor lesznek kölcsönösen kizáróak, ha különböző köztes helyről szállítanak, és a szállítási időben szintén van közös rész. Ezek mellett egyes specifikusabb problémák esetén további kölcsönösen kizáró tevékenységeket is találhatunk csupán a struktúra, és a feladatleírás vizsgálatával. Az előzőekben felsorolt esetek azok voltak, melyek ha megjelennek, biztosan kölcsönös kizárást vonnak maguk után.

Szemléltetésképpen egy olyan VAP feladatot mutatunk be, melyben három elvégzendő feladat van (P1, P2, P3), és ehhez három különböző jármű áll rendelkezésre (R1,R2,R3) ([6]). A feladat leírása alapján a VPA-PNS modellkonverziós eljárás 14



4.7. ábra. A kölcsönös kizárás kezelését szemléltető jármű hozzárendelési probléma maximális struktúrája

darab tevékenységet azonosított. A generált PNS feladat maximális szuperstruktúráját a 4.7 ábrán láthatjuk.

Az előzőekben bevezetett szempontok alapján azonosíthatóak a kölcsönösen kizáró tevékenység halmazok. Az $\{o_1, o_2\}$ és az $\{o_3, o_4, o_5\}$ halmazok azért lesznek kölcsönösen kizáróak, mert páronként közös kiindulási helyről indulnak (ugyanaz a jármű) és olyan időpontokban kell megérkezniük, hogy az úton töltött időintervallumoknak lenne metszete, így az adott jármű egyszerre nem lehet több helyen. A $\{P3byR2, P3byR1\}$, a $\{P1byR2, P1byR1\}$, valamint a $\{P2byR1, P2byR3\}$ tevékenység halmazok azért lesznek kölcsönösen kizáróak, mert ugyanazt a feladat megvalósítását modellezik, és egy feladatot egyszer kell csak elvégezni. Továbbá kölcsönösen kizáróak lesznek az $\{o_1, o_7\}$ és az $\{o_4, o_8\}$ halmazok is, mivel ezek olyan kettes csoportba tartozó tevékenységek, amelyek ugyanazt a járművet mozgatják, és célállomásuk, valamint odaérkezési idejük megegyezik és ha egyszerre szerepelnének, akkor az adott járműnek egyszerre több helyen kellene lennie, ami ellentmondás.

Összesen hét kölcsönösen kizáró műveleti egység halmaz azonosítható a példában. Az alkalmazott PNS modellt kiegészítettük velük. Ezt követően újra meg lett oldva a probléma.

4.1. táblázat. A jármű hozzárendelési tesztfeladat eredményéül kapott hozzárendelések és a megoldás során generált részproblémák száma

Megoldás	**VAP-PNS modell**			**Kölcsönös kizárással kiegészítve**		
#	P1	P2	P3	P1	P2	P3
1	R1	R3	R1	R1	R3	R1
2	R1	R3	R2	R1	R3	R2
3	R2	R1	R2	R2	R1	R2
4	R2	R3	R2	R2	R3	R2
5	R2	R3	R1	R2	R3	R1
Részproblémák [darab]	173			45		

A feladat első öt legjobb megoldását határoztuk meg az alap VAP-PNS modellre (ahogyan a szakirodalomban is szerepel), valamint a kölcsönös kizárással kiegészített modellre is a kölcsönös kizárást figyelembe vevő PNS megoldási módszer alapján. A megoldások eredményei a 4.5 táblázatban láthatóak. Az egyes sorok a számított jármű hozzárendeléseket tartalmazzák külön-külön a módszerekhez, azaz, hogy mely Pi szállítást mely Ri jármű fogja végezni (i = 1, 2, 3). Mind az öt megoldás megegyezik, azonban a megoldás során generált részproblémák számában jelentős eltérés tapasztalható. Az eredeti módszer, amely nem képes a kölcsönösen kizáró halmazok alapján elvetni részproblémákat, 173 részprobléma vizsgálatával oldotta meg a feladatot. Míg az új, kölcsönös kizárást figyelembe vevő módszernek a kibővített modell alapján sikerült ugyanazon eredményeket csupán 45 részprobléma vizsgálatán keresztül szolgáltatnia, azaz az eredeti módszer által bejárt keresési tér méretének 26%-ának vizsgálata szükséges mindössze a megoldások generálásához.

4.6. További feladatok futtatási eredményei

Ahogy azt az előző fejezetben láthattuk a jármű hozzárendelési feladat esetén a kölcsönös kizárások kezelése drasztikusan csökkenti azon feladatok megoldásához bejárandó keresési tér méretét, amelyek kölcsönösen kizáró tevékenységeket tartalmaznak. A most következő fejezetben további kettő a szakirodalomban publikált reakcióút azonosítás (Reaction Pathway Identification, RPI) témakörébe tartozó feladat futási eredményeivel szemléltetem a PNS módszertan kölcsönös kizárások kezelésével kibővített változatának a jelentőségét.

A reakcióutak meghatározása elengedhetetlen, ha kémiai vagy biokémiai reakciók mechanizmusának kinetikáját kívánjuk megismerni. A reakcióút elemi reakciók lépéseiből áll, melyek a reakció előfeltételeitől (a reagensektől) elvezetnek a végtermékig. Az eredő reakciót az elemi reakciólépések egy hálózata építi fel. Az elemi reakciólépések adott arányban használnak fel és állítanak elő kémiai vagy aktív átmeneti részecskéket. Ennek köszönhetően, az elemi reakciók hálózatát pontosan meghatározza a hálózatban szereplő egyes elemi reakcióknak az eredő reakcióhoz viszonyított relatív sebessége ([14]). Mivel minden elemi reakció megfordítható, az előre és a fordított lépést is magában foglalja. A feladat formális megadása során minden elemi reakció előre és fordított irányú lépésként is szerepel a lehetséges lépések halmazában. Az oda- és visszirányú elemi reakciólépések együttese nulla eredőjű kört alkotnak. Bertók definiálta a lehetséges reakcióutak azonosítására szolgáló 5 axiómát, amelyek közül az R5 axióma kimondja, hogy "A reakcióutat reprezentáló hálózat nem tartalmaz nulla eredőjű kört." ([14], [35]). Azaz egy reakcióhálózat egyszerre nem tartalmazhatja az oda- és visszirányú lépések közül mindkettőt, tehát kölcsönösen kizáró halmazokat alkotnak.

Észrevehetjük, hogy a reakcióút azonosítás feladat (RPI) elemi reakciólépései a PNS módszertan tevékenységeivel, továbbá az elemi reakciólépések ki- és bemeneti anyagáramai pedig a tevékenységek ki- és bemeneti entitásaival azonosíthatóak. Azaz a PNS módszertan algoritmusai segítségével az előzőekben bemutatott RPI feladatok is megoldhatóak.

Első példaként a bután (C_4H_{10}) buténné (C_4H_8) való dehidrogénezése szolgál, ahol

4.2. táblázat. Futtatási eredmények kiértékelése

Feladat megnevezés	**Kölcsönös kizárás nélkül vizsgálandó struktúrák száma [darab]**	**Kölcsönös kizárással vizsgálandó struktúrák száma [darab]**
RPI (Bután)	12	3
RPI (Ammónia)	3840	17

adott az eredő reakció ($C_4H_{10} \rightarrow C_4H_8 + H_2$), az elemi reakcióknak pedig a Temkin által javasolt halmazát ([105]) tekintjük. A feladat 4 elemi reakció lépést tartalmaz és 2 közvetlen reakcióútja van.

Második feladatként azt a 11 elemi reakciót tartalmazó ammónia-szintézist ($N_2 + 3H_2 \rightarrow 2NH_3$) tekintjük, amelyhez tartozó összes közvetlen utat a szakirodalomból ismerhetjük ([54], [55]). A feladatnak 17 közvetlen reakcióútja van.

Mindkét feladat közvetlen reakcióútjainak mindegyike generálható a PNS módszer alap algoritmusaival és a kölcsönös kizárással kiegészített változatokkal is. Mindegyik megoldás megegyezik, azonban a megoldás során generált részproblémák számában jelentős eltérés tapasztalható, ahogy azt a 4.2 táblázat jól szemlélteti. Az eredeti módszer, amely nem képes a kölcsönösen kizáró halmazok alapján elvetni részproblémákat, az első feladat (bután buténné való dehidrogénezése) esetén 12, míg a második feladat esetén (ammónia-szintézis) 3840 részprobléma vizsgálatával oldotta meg a feladatot. Az új, kölcsönös kizárást figyelembe vevő módszernek a kibővített modell alapján sikerült ugyanazon eredményeket csupán 3 illetve 10 részprobléma vizsgálatán keresztül szolgáltatnia, azaz az eredeti módszer által bejárt keresési tér méretének 25%-ának, illetve 0,44%-ának vizsgálata szükséges mindössze a megoldások generálásához.

4.7. Összefoglalás

Bemutattam milyen megfontolások miatt szükséges a PNS keretrendszer kiterjesztése a kölcsönösen kizáró halmazok kezelésére, és hogy milyen változtatásokat kellett

elvégezni az eredeti P-gráf keretrendszeren, hogy annak algoritmusai hatékonyan figyelembe tudják venni ezeket az új megkötéseket.

A P-gráf módszertanba bevezettem egy kiegészítő strukturális feltételt a meglévő öt alap axióma mellé, amely biztosítja, hogy egyetlen egy megoldás struktúra sem tartalmaz kölcsönösen kizáró tevékenységeket. Az új korlátozások bevezetése keresési tér csökkenést eredményezhet, melyet az algoritmusok bemutatott módosításával hatékonyan ki lehet használni a megoldási idő csökkentésére, valamint a megoldás során generált részproblémák számának csökkentésére. Az ehhez szükséges algoritmus módosításokat a dolgozatomban bemutattam, amely révén megszülettek az SSGMX és az ABBMX algoritmusok. Az új módszer működését a szakirodalomból vett járműhozzárendelési problémán mutattam be, amely megoldása során szignifikáns gyorsulást sikerült elérni.

Ezidáig csak érintőleges vizsgálatok történtek a kölcsönös kizárásnál komplexebb strukturális megkötések – például OCL kényszerek validációja – bevonására a P-gráf módszertannal történő algoritmikus szintézis folyamatába. Igaz ezen vizsgálatok nem tárgyai a jelen dolgozatnak, de érdemes említést tenni róluk, abból a szemszögből, hogy ez a terület egy igen ígéretes továbblépési lehetőség lehet, amelyre a későbbiekben majd nagyobb hangsúlyt kell fektetni.

4.8. Kapcsolódó publikációim

Nemzetközi konferencia előadások

2. Tarczali, T., Kadanóczki. H., Ősz, E., Zachár, G., Kalauz, K., and Süle, Z., P-graph Representation and Structural Analysis of Business Process Models, VOCAL 2010, Veszprém, Hungary, December 13-15, 2010.

1. Kalauz, K., and Bertók, B., Algorithmic Synthesis of Structural Alternatives for Business Processes, CSCS 2010, Szeged, Hungary, June 29-July 2, 2010.

5. fejezet

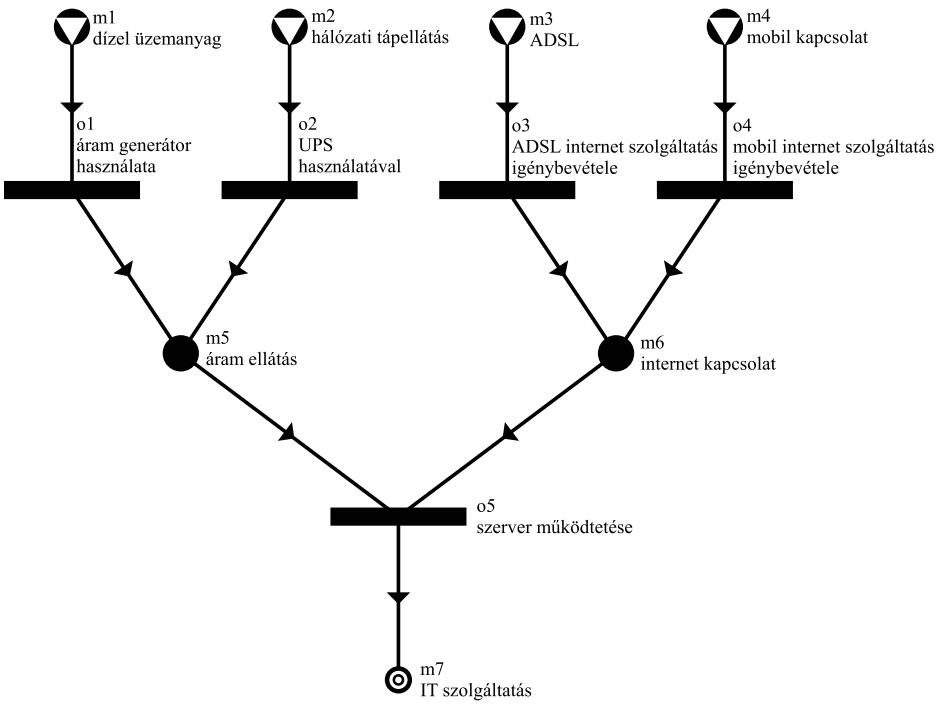
A redundáns folyamat-hálózatok szerepe és fontossága

A most következő fejezetben a redundáns folyamat-hálózatok szerepe és fontossága, a redundáns hálózatok generálására szolgáló algoritmus, továbbá azok megbízhatóságának számítási módszere kerül bemutatásra.

5.1. A redundáns folyamat-hálózatok generálásának motivációja

A 5.1. ábrán látható motivációs példa szolgál annak bemutatására, hogy miért lehet hasznos a folyamat-hálózatokat a redundancia szemszögéből vizsgálni, illetve redundáns esetekkel bővíteni. Egy IT szolgáltatás biztosítását szemléltető példa leegyszerűsített változatát vettük alapul, ahol is kiemelt fontossággal bír az áram ellátás illetve az internet szolgáltatás biztosítása, ahhoz hogy a szerver zavartalanul ki tudja szolgálni a felhasználókat.

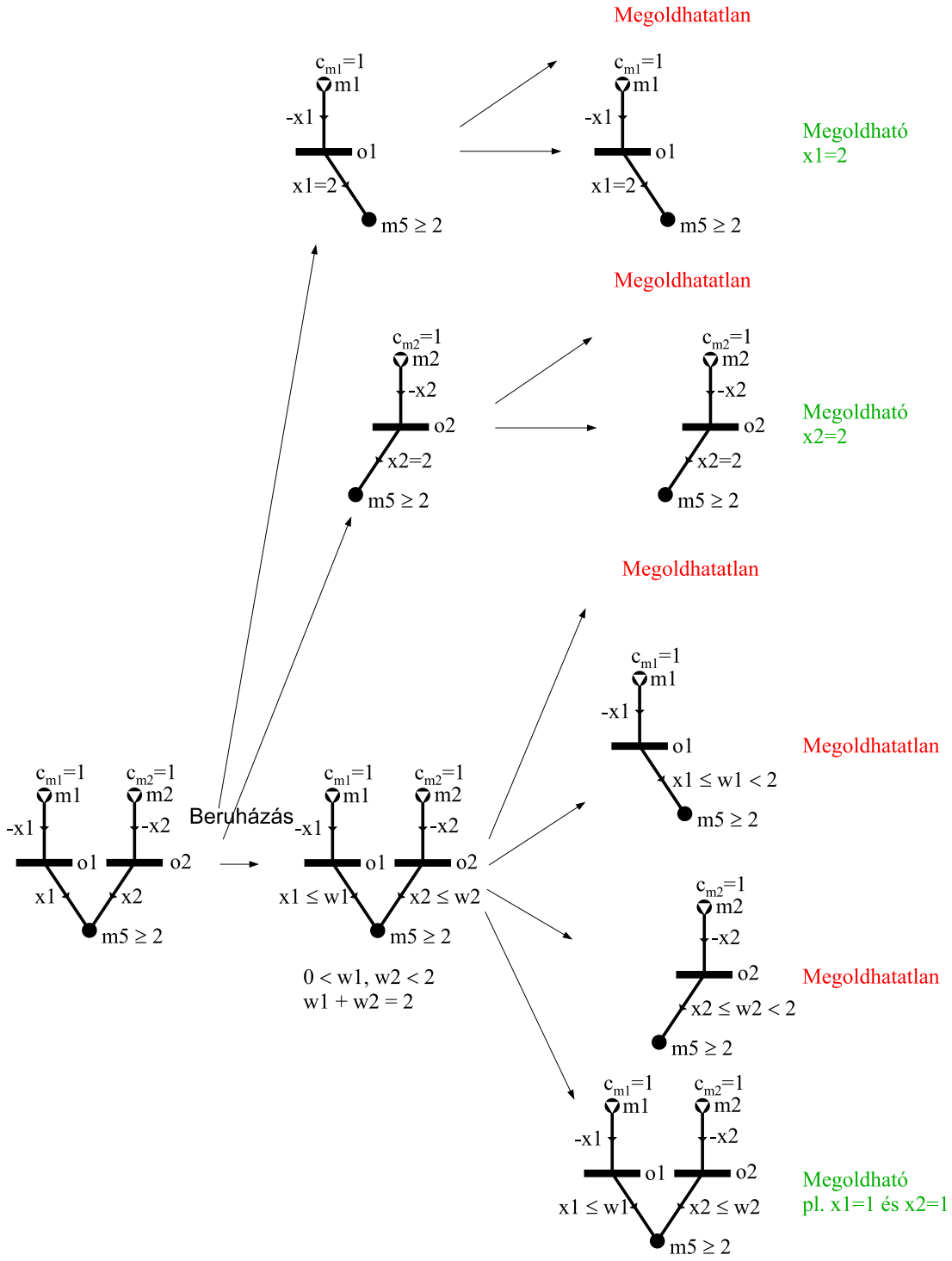
Az imént bemutatott feladatnak a baloldali áram ellátást biztosító részét használjuk fel a redundancia fontosságának bemutatására. A kiválasztott példa egy nagyon egyszerű esetet szemléltet, amely a következő. Két erőforrás az $m1$ (dízel üzemanyag) és az $m2$ (hálózati betáplálás) szerepel a feladatban, amelyek c_{m1} és c_{m2} költségei egységenként 1 és amelyeket egy-egy tevékenység az $o1$ (áram generátor használata) és



5.1. ábra. Egy IT szolgáltatás folyamatát bemutató példa

az $o2$ (UPS használatával) használhat fel. Mindkét tevékenység az $m5$ (áram ellátás) entitást képes előállítani egy az egy arányban az $m1$ és $m2$ erőforrásokból. A feladatban szerepel egy megkötés a végcélra, mégpedig az hogy minimum 2 egységnyit kell előállítani belőle. Azaz vagy hálózati betáplálás vagy áram generátor segítségével esetleg a kettővel egyszerre egy bizonyos áramszintet mindig biztosítani kell a zavartalan működéshez. A továbbiakban mind a tevékenységekre mind pedig az entitásokra a rövid neveikkel utalunk majd.

A kiragadott részfeladat elvégzéséhez szükséges beruházás során kombinatorikusan három lehetőség áll rendelkezésünkre, amelyek a következők. Az első és második lehetőség lényegében megegyezik, csak abban különböznek, hogy egyszer az egyik tevékenység, másszor a másik tevékenység szerepel benne. A lényege az, hogy csak az egyik tevékenység ellátására történik beruházás akkora kapacitással, hogy az egymaga képes legyen előállítani az $m5$ entitást azaz a megfelelő áramszintet. Ezeknek az eseteknek kombinatorikusan két-két további esete lehet. Az első eset az, amikor az adott tevékenység nem szerepel a struktúrában, és ezáltal ez a megoldás megoldhatatlan. A

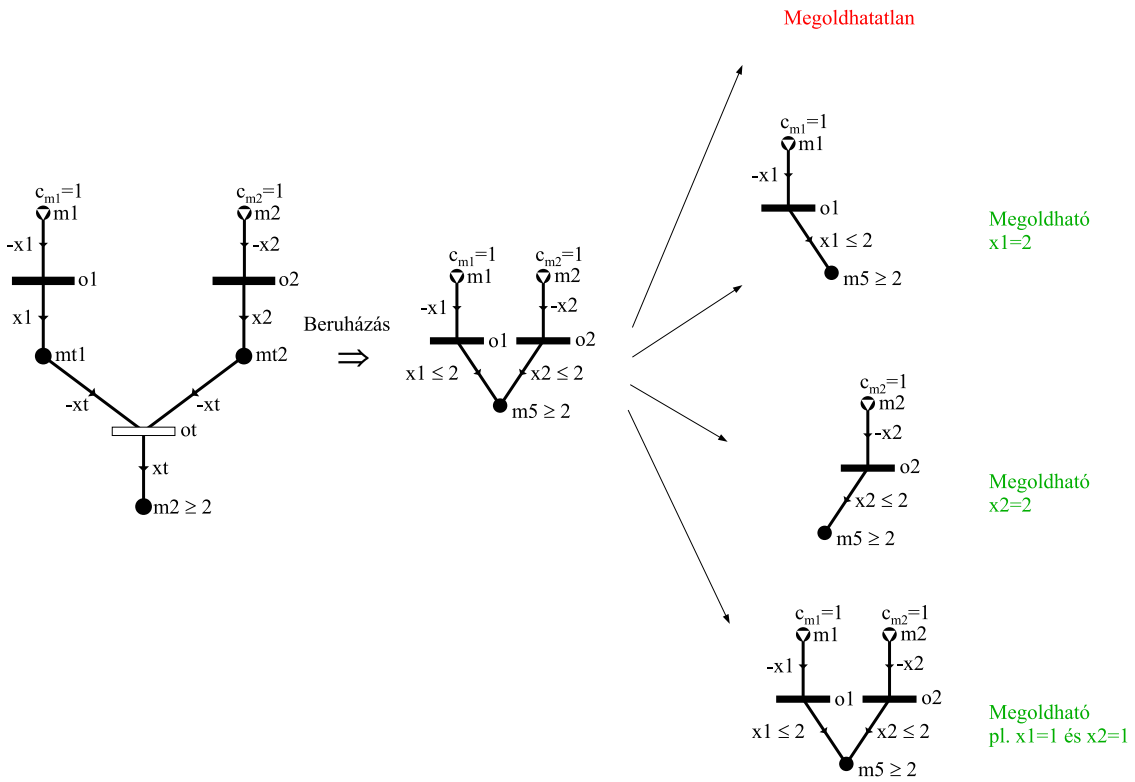


5.2. ábra. Az alap motivációs példa

második az, amikor szerepel a struktúrában az adott tevékenység a megfelelően nagy ($x1 \leq 2$, illetve $x2 \leq 2$) kapacitás korlátok mellett, azaz az $o1$ tevékenység $x1 = 2$, és az $o2$ tevékenység pedig $x2 = 2$ terheléssel.

Vizsgálatunk során a harmadik lehetőség lesz az, amit majd a későbbiekben tovább vizsgálunk. Ebben az esetben mindkét tevékenységre megtörténik a beruházás. Viszont ahhoz, hogy az $m5$ a kívánt mennyiségben előálljon, kisebb kapacitás is elég mindkét tevékenység esetében ($x1$ az $o1$, $x2$ pedig az $o2$ tevékenység kapacitása), mintha egymagukban kellene előállítani a megfelelő áramellátást, azaz $x1 \leq w1$ és $x2 \leq w2$, ahol $0 < w1, w2 < 2$ és $w1 + w2 = 2$. Kombinatorikusan négy további eset állhat elő a feladat megoldása során. Az első eset, amikor nem szerepel egyik tevékenység sem a struktúrában. Ebben az esetben értelemszerűen az $m5$ nem tud előállni, így megoldhatatlan a feladat. A második eset az, amikor csak az $o1$ tevékenység szerepel a struktúrában, amely a korábbi megállapítás alapján a $x1 = w1 < 2$ maximális terheléssel sem képes a megkövetelt mennyiségű $m5$ entitást, azaz áramot előállítani, így ebben az esetben is megoldhatatlan a feladat. A harmadik eset, az amikor csak az $o2$ tevékenység szerepel a struktúrában, amely a $x2 = w2 < 2$ maximális terheléssel sem képes a megkövetelt mennyiségű végcélt előállítani, így ebben az esetben is megoldhatatlan a feladat. A negyedik eset az, amikor mind az $o1$ és mind az $o2$ tevékenység szerepel a struktúrában. Ebben az esetben megoldhatóvá válik a feladat akkor, ha mindkét tevékenység szerepel a struktúrában a maximális kapacitásukkal, azaz $x1 = w1$ és $x2 = w2$, mivel $w1 + w2 = 2$. Megoldás lehet például az $x1 = 1$ és $x2 = 1$ terhelésű tevékenységekkel rendelkező struktúra. Tehát abban az esetben, ha egyik tevékenység sem tudja önállóan előállítani a végcélt, akkor csak akkor megoldható a feladat, ha mindkét tevékenység szerepel a megoldásban a megfelelő kapacitással. Az részfeladat előzőekben bemutatott leírása a 5.2 ábrán látható. Az egyes kombinatorikusan lehetséges struktúrák az ábra jobb oldalán fentről lefelé találhatóak meg.

Ahogy láthattuk az előzőekben bemutatott részfeladatban a harmadik lehetőség esetén – amikor mindkét tevékenység elvégezésére megtörténik a beruházás –, ha kiesik az egyik tevékenység a folyamatból, akkor a feladat megoldhatatlanná válik. Tehát a folyamat függ mindkét tevékenység sikeres végrehajtásától. Ha növelni szeretnénk a



5.3. ábra. A motivációs példa redundanciával kibővített változata

folyamat robusztusságát, a hibatűrő képességét, akkor minél jobban törekednünk kell arra, hogy legyenek alternatív szcenáriók a folyamatban az esetleges meghibásodások és egyéb nem várt események kezelésére. Az előzőekben bemutatott részfeladatot redundáns esetekkel kibővítve egy sokkal hibatűrőbb hálózatot kapunk. Ez a bővített feladat a 5.3 ábrán látható. Az alaphálózatba bekerült az *ot* tevékenység és az integrálásához szükséges $mt1$ és $mt2$ entitások. Az *ot* nem egy a hagyományos értelemben vett tevékenység, hanem egy olyan tevékenység, amely megköveteli a bejövő ágakon elhelyezkedő tevékenységektől, hogy a kapacitásaikat felemeljék olyan szintre, hogy önmagukban is eredményezni tudják a végcélt. Így az optimális beruházás eredményeként mindkét tevékenység kapacitása 2, vagyis az $o1$ tevékenység kapacitása $x1 \leq 2$ és az $o2$ kapacitása $x2 \leq 2$ korlátokkal indul neki a folyamat végrehajtásának. Ennek a változatnak is kombinatorikusan négy esete lehet, ahogy azt már az előzőekben is láttuk. Az első eset az ha az üzemeltetés során mindegyik tevékenység kiesik a struktúrából. Ebben az esetben értelemszerűen az $m5$ nem tud előállni, így csakúgy mint az alapfeladatban megoldhatatlan a feladat. A második eset az, amikor csak az $o1$ tevékenység szerepel a struktúrában. Mivel az $o1$ tevékenység $x1 \leq 2$ kapacitása megfelelően nagy lett a *ot* tevékenység által, ezért ez a lehetőség megoldható $x1 = 2$ terhelés esetén. A harmadik eset az, amikor csak az $o2$ tevékenység szerepel a struktúrában. Mivel az $o2$ tevékenység $x2 \leq 2$ kapacitása is megfelelően nagy lett a *ot* tevékenység által, ezért ez a lehetőség is megoldható $x2 = 2$ terhelés esetén. Végül a negyedik eset az, amikor mind az $o1$ és az $o2$ tevékenység szerepel a struktúrában. Mivel mindkét tevékenység megfelelően nagy kapacitással rendelkezik ($x1 \leq 2$ és $x2 \leq 2$), ezért széles skálán mozog a megoldások száma. Megoldás lehet például ha az $o1$ tevékenységnek $x1 = 1$, az $o2$ tevékenységnek pedig $x2 = 1$ a megengedett terhelése.

Tehát miután redundánssá vált a hálózat, sokkal robusztusabb lett a folyamat. Míg az alapfeladatban, ha kiesik valamelyik tevékenység, a feladat megoldhatatlanná válik, azaz négy esetből csak egynek létezik megoldása, addig a kiegészített változatban, ha legalább egy tevékenység működőképes, akkor teljesülhet az $m5$ entitás megfelelő mértékű előállítása – a megfelelő áram szint biztosítása –, azaz a négy esetből háromnak létezik megoldása. Hasonló módon vizsgálhatnánk meg 5.1. ábrán

található hálózat jobboldali internet kapcsolatot biztosító részét, amely vizsgálattól most eltekintünk, mivel a struktúrája pontosan megegyezik a korábban bemutatottal.

5.2. A redundancia szerepe és fontossága

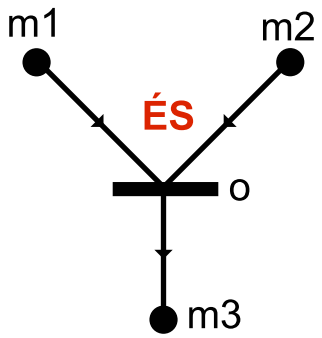
Mivel a végcélok gyakran több különböző módon is elérhetőek ezért az alternatív szcenáriók, például az ellátási lánc struktúrák, megbízhatóságának és költségeinek elemzése magas gyakorlati hasznossággal bír ([26], [51], [36], [56], [85]). Az ellátási struktúrában, minél többféle a végcélok eléréséhez elegendő láncelemek kombinációja, annál nagyobb az ellátási lánc teljes megbízhatósága, azaz a robusztussága. A megbízhatóság mellett az ellátási struktúrák összköltsége is széles tartományban változhat. Mivel a strukturális alternatívák teret adhatnak a redundanciának, és ily módon meg van a lehetősége a megbízhatóság növekedésének, ezért a megbízhatóság nézőpontjából a strukturális alternatívák logikus értelmezése megköveteli az utólagos figyelmet.

A szintézis feladatokban, és az eredményül kapott ellátási szcenáriókban szereplő tevékenységek logikai kapcsolatainak egyértelmű tisztázására a P-gráf módszertan egy jól megalapozott matematikai modellt szolgáltat, ami elengedhetetlen ahhoz, hogy eldönthető legyen egy szcenárióról, hogy az redundáns-e vagy sem.

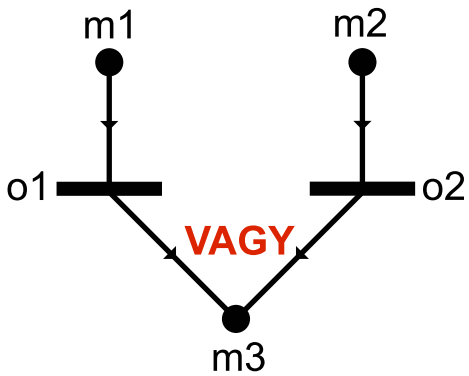
Lemma 5.2.1 *Ha egy tevékenységnek több, mint egy előfeltétele van, akkor mindegyik előfeltétel szükséges a tevékenység véghezviteléhez, ami egy logikai „ÉS” kapcsolatnak felel meg.*

Ha egy tevékenységnek több, mint egy előfeltétele van, akkor a tevékenység csomópontot a P-gráf reprezentációban egy olyan vízszintes téglalappal jelöljük, aminek ugyanannyi bementi éle van mint amennyi előfeltétele. Felhasználva a 5.2.1 lemmát a 5.4. ábrán látható P-gráf esetében az $m1$ „ÉS” az $m2$ egyaránt szükséges előfeltételei az o tevékenységnek.

Lemma 5.2.2 *Ha egy entitás kettő vagy több tevékenység által is elérhető, akkor ezen tevékenységek bármely kombinációja elégséges lehet, ami egy logikai „VAGY” kapcsolatnak felel meg.*



5.4. ábra. A logikai „ÉS” kapcsolat P-gráf reprezentációja

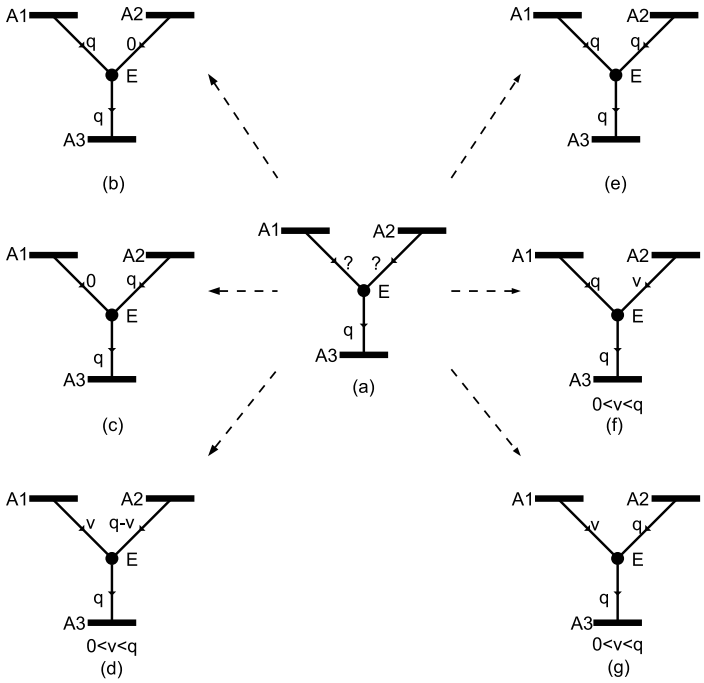


5.5. ábra. A logikai „VAGY” kapcsolat P-gráf reprezentációja

Amikor egy entitás előállhat több mint egy tevékenység kimeneteként, akkor azt az entitás csomópontot a P-gráf reprezentációban egy olyan kitöltött körrel jelöljük, aminek több bementi éle van. Felhasználva a 5.2.2 lemmát a 5.5 ábrán látható P-gráf esetében az „Eredmény” előállhat az $o1$ „VAGY” az $o2$ tevékenység által. Az előzőekben bemutatott körülmények alapján három olyan alternatív szcenárió állítható elő, amivel elérhető az $m3$ entitás. Az első szcenárió az, amikor kizárólag az $o1$ tevékenységet, a második, amikor kizárólag az $o2$ tevékenységet, a harmadik pedig amikor párhuzamosan mind az $o1$ és mind az $o2$ tevékenységeket felhasználva jutunk el az $m3$ entitásig.

A rendszer megbízhatóság elmélete két fő struktúrát vesz figyelembe. Ezek a soros és a párhuzamos rendszerek ([26], [51], [36], [56], [85]). A soros rendszerek azokat a struktúrákat szimbolizálják, ahol minden egyes tevékenységet sikeresen és

sorrend helyesen végre kell hajtani ahhoz, hogy elérjük a végcélt. Ily módon a soros rendszerek teljes megbízhatósága a sorban található tevékenységek egyedi megbízhatóságainak az eredménye, azaz a megbízhatóság tipikusan csökken egy új tevékenység a sorozatba való bevételével. A párhuzamos rendszerek olyan struktúrákat írnak le, ahol a végcél elérhető, ha a párhuzamos tevékenységek közül legalább egy sikeres. Tekintsük a teljes és a valós megbízhatóság különbségét a kockázatnak, azaz $Kockázat = 100\% - megbízhatóság$. A párhuzamos rendszer teljes kockázata a párhuzamosan végrehajtható tevékenységek egyéni kockázatainak az eredménye, azaz a teljes kockázat csökkenthető újabb párhuzamos tevékenységek bevonásával.



5.6. ábra. Két tevékenység együttműködésének alternatív strukturális reprezentációi

Azokban a komplex folyamathálózatokban, ahol a tevékenységek nem redundánsak, a teljes megbízhatóság a soros rendszerekhez hasonlóan számítható ki, azaz az összes tevékenységnek sikeresnek kell lennie figyelmen kívül hagyva a hálózatok topológiáját.

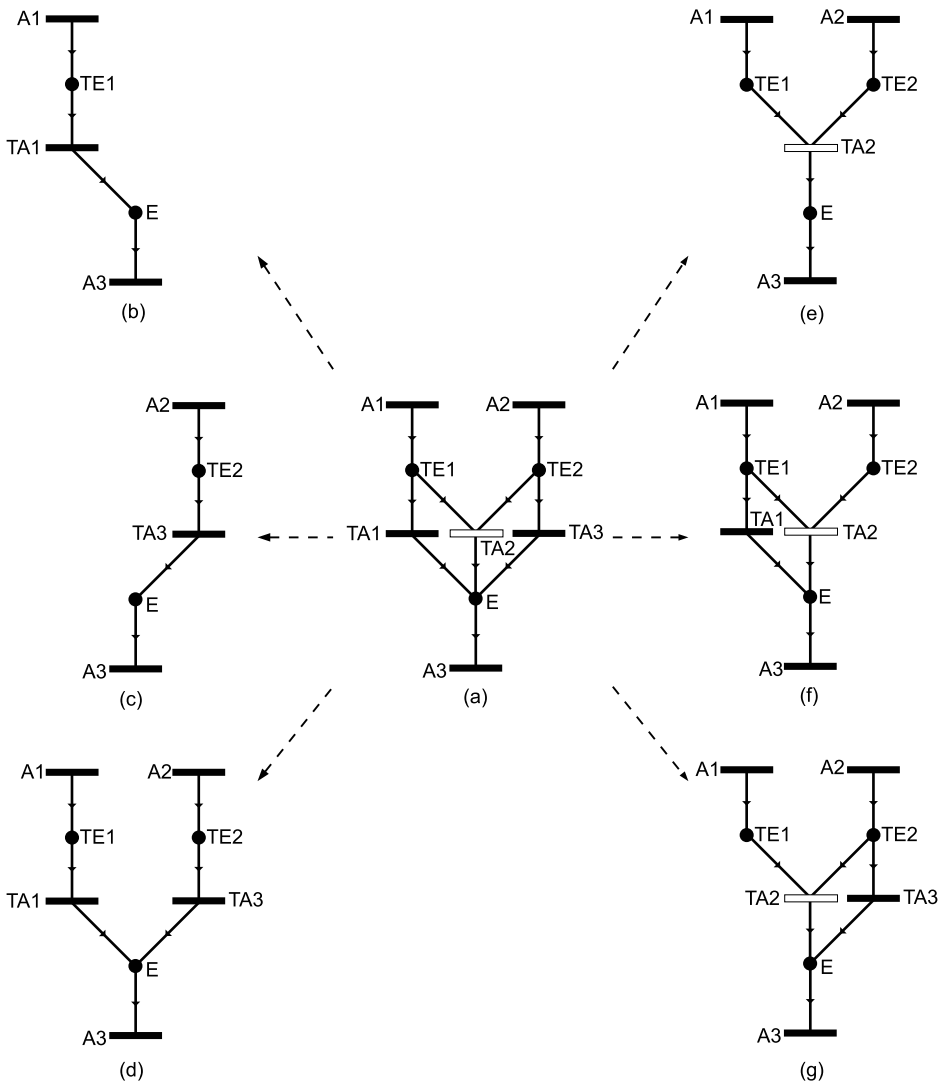
Az előzőekben bemutatott VAGY kapcsolat kizárólag kapacitás korlát nélküli strukturális feladatok esetén alkalmazható, azaz abban az esetben, ha csak az a kérdés,

hogy egy tevékenység szerepel-e a struktúrában vagy sem. Viszont ha a tevékenységek rendelkeznek kapacitás korlátokkal, amiket figyelembe kell venni a megoldás generálásnál, akkor az előzőekben bemutatott VAGY kapcsolat számos különböző lehetőséget eredményezhet. Például két tevékenység VAGY kapcsolata esetén hat lehetséges szcenárió különböztethető meg, ahogy az az 5.6 ábrán látható. Az 5.6.a ábra a kiindulási állapot, amelyből aztán hatféle eset állhat elő, amelyek a következők. Az 5.6.b ábrán egyedül az A1 tevékenység eredményezi az E entitást, az A2 tevékenység pedig nem vesz részt a folyamatban. Az 5.6.c ábrán egyedül az A2 tevékenység eredményezi az E entitást, az A1 tevékenység pedig nem vesz részt a folyamatban. Az 5.6.d ábrán az A1 és az A2 tevékenység együttesen eredményezik az E entitás megkövetelt mennyiségét. Az 5.6.e ábrán mind az A1 és mind az A2 tevékenység biztosítja a szükséges mennyiséget az E entitásból, azaz redundánsan párhuzamosak. Az 5.6.f ábrán az A1 tevékenység a teljes q mennyiséget, míg az A2 tevékenység csak egy kisebb v mennyiséget eredményez az E entitásból, vagyis csak részben redundánsak. Végül az 5.6.g ábra azt a szcenáriót szemlélteti, amikor az A2 tevékenység a teljes q mennyiséget, míg az A1 tevékenység csak egy kisebb v mennyiséget eredményez az E entitásból, vagyis szintén csak részben redundánsak.

Ahhoz, hogy egyértelműen megkülönböztethető legyen az 5.6 ábrán bemutatott hat szcenárió, a strukturális reprezentációt ki kell bővíteni olyan kiegészítő entitásokkal és tevékenységekkel, amelyek a folyamat karakterisztikáján nem változtatnak, viszont a segítségükkel egyértelműen meghatározható mind a hat eset, ahogy az az 5.7 ábrán látható. Ilyen kiegészítő entitások a TE1 és a TE2, és kiegészítő tevékenységek a TA1, TA2 és TA3.

A TA2 tevékenység – amelyet egy fehér téglalap jelöl, hogy vizuálisan megkülönböztethető legyen – nem egy, a hagyományos értelemben vett tevékenység, hanem olyan, amely megköveteli a bejövő ágakon elhelyezkedő tevékenységektől, hogy a kapacitásaikat felemeljék olyan szintre, hogy önmagukban is eredményezni tudják a TA2 kimeneteit, azaz jelen esetben az E entitást. A továbbiakban a hasonlóan jelölt tevékenységek mindegyikére igaz az előzőleg megállapított tulajdonság.

Az 5.7.a ábra szemlélteti a redundánsan kibővített szuperstruktúrát, amelyből



5.7. ábra. Két párhuzamos tevékenység alternatív szcenárióinak szuperstruktúrája

aztán a következő hat eset állhat elő. Az 5.7.b ábrán egyedül az A1 tevékenység eredményezi a TE1 entitáson és TA1 tevékenységen keresztül az E entitást, az A2, a TA2 és a TA3 tevékenységek és a TE2 entitás pedig nem vesznek részt a folyamatban. Az 5.7.c ábrán egyedül az A2 tevékenység eredményezi a TE2 entitáson és TA3 tevékenységen keresztül az E entitást, az A1, a TA1 és a TA2 tevékenységek és a TE1 entitás pedig nem vesznek részt a folyamatban. Az 5.7.d ábrán az A1 és az A2 tevékenységek együttesen biztosítják a TE1 és TE2 entiások és a TA1 és TA3 tevékenységek közreműködésével a szükséges mennyiséget az E entitásból, a TA2 tevékenység pedig nem vesz részt a folyamatban. Az 5.7.e ábrán az A1 és az A2 tevékenységek mindegyike eredményezi a TE1, a TE2 entitások és a TA2 tevékenység közreműködésével az E entitás megkövetelt mennyiségét, azaz redundánsan párhuzamosak, a TA1 és a TA3 tevékenységek pedig nem vesznek részt a folyamatban. Az 5.7.f ábrán az A1 tevékenység a teljes mennyiséget, míg az A2 tevékenység csak egy kisebb mennyiséget eredményez a TE1 és TE2 entitások valamint a TA1 és TA2 tevékenységek közreműködésével az E entitásból, vagyis csak részben redundánsak, a TA3 tevékenység pedig nem vesz részt a folyamatban. Végül pedig az 5.7.g ábra azt a szcenáriót szemlélteti amikor az A2 tevékenység a teljes mennyiséget, míg az A1 tevékenység csak egy kisebb mennyiséget eredményez a TE1 és TE2 entitások valamint a TA2 és TA3 tevékenységek közreműködésével az E entitásból, vagyis szintén csak részben redundánsak, a TA3 tevékenység pedig nem vesz részt a folyamatban.

Fontos megjegyezni, hogy a P-gráf 5.7 ábrán bemutatott kiterjesztése az 5.6 ábrán bemutatott összes szcenárió egyértelmű és egyedi strukturális reprezentációja.

5.3. Redundancia generálás

Az 5.8 ábrán látható redundancia generáló RG (Redundancy Generation) algoritmus a maximális struktúra kiegészítésére lett kidolgozva annak érdekében, hogy ki lehessen aknázni az előzőekben bemutatott, a redundanciában rejlő lehetőségeket ([13], ([65], [63], [64])). Az algoritmus az 5.6 ábra és az 5.7 ábra közötti átalakítás általánosítása tetszőleges bemenetű entitásokra.

Definíció 5.3.1 *A kettő vagy több bemeneti éllel rendelkező entitásokat redundánsan kiterjeszthető entitásoknak nevezzük.*

Tétel 5.3.1 *Az erőforrás típusú ($\mathcal{R}$) entitásokat nem lehet redundánsan kiterjeszteni.*

Bizonyítás *Mivel az erőforrás típusú entitásoknak nincs bementi élük – ahogy azt a S2-es PNS axióma mutatja – és mivel a 5.3.1 definíció kimondja, hogy legalább kettő bementi él szükséges a redundáns kiterjesztéshez, ezért az erőforrás típusú entitások nem terjeszthetőek ki redundánsan.*

Az RG algoritmus indulásakor az 1. lépésében összegyűjti a *split* halmazba a szuperstruktúra minden olyan entitását, amelynek több mint egy tevékenység a következménye. Ha nem talál ilyet, azaz a feladat nem tartalmaz redundánsan kiterjeszthető entitást, akkor befejeződik az algoritmus futása. Ha viszont talál, azaz a *split* halmaz nem üres, akkor ezután a 2. lépés **while** ciklusának minden iterációjában kiválaszt a halmazból egy még nem vizsgált x entitás típusú csúcsot, amelyre elvégzi a redundáns átalakítást. Azaz a 3. lépésben számba veszi, hogy hány tevékenység következménye az x entitás, megszünteti az entitás és a tevékenységek közötti kapcsolatot, és a tevékenységekhez hozzáköti a kiegészítő entitásokat, amelyek majd a kiegészítő tevékenységek bemenetéül szolgálnak. Végül a 4. lépésben hozzáadja a szuperstruktúrához azokat a kiegészítő tevékenységeket, amelyek megteremtik a kapcsolatot a kiegészítő entitások és az x entitás között, és amelyek segítségével aztán az összes redundancia variáció előállítható.

Definíció 5.3.2 *Egy tetszőleges struktúra redundánsan kiterjesztett változata egy olyan struktúra, ahol bizonyos entitásokra megköveteljük, hogy a tevékenységek valamely kombinációja együttesen, azaz redundánsan eredményezze.*

Tétel 5.3.2 *Az RG algoritmus által generált redundánsan kiterjesztett maximális struktúra a PNS feladat maximális struktúráinak és azok lehetséges redundánsan kiterjesztett változatainak uniója.*

```
01.        function RG((m, o): maximal structure):
                                redundancy extended maximal structure
02.        begin
03.            count := 0;
04. 1. lépés   split := ⋃_{m''∈(ψ+(o)∩ψ−(o))∪P and |φ−(m'')|>1} {m''};
05. 2. lépés   while split ≠ ∅ begin;
06.                m' := ∅;
07.                x ∈ split;
08.                o_out := φ−({x});
09. 3. lépés       while o_out ≠ ∅ begin;
10.                    (α, β) ∈ o_out;
11.                    o_out := o_out \ {(α, β)};
12.                    count := count + 1;
13.                    m' := m' ∪ {x_count};
14.                    o := o \ {(α, β)} ∪ {(α, β \ {x} ∪ {x_count})};
15.                end;
16.                TEP := ℘(m') \ {∅};
17. 4. lépés       for all γ ∈ TEP begin;
18.                    o := o ∪ {(γ, {x})};
19.                end;
20.                m := m ∪ m';
21.                split := split \ {x};
22.            end;
23.        end;
```

5.8. ábra. A redundancia generáló RG algoritmus

Megjegyzés 5.3.1 *A RG algoritmus által generált redundánsan kiterjesztett maximális struktúra akkor és csak akkor különbözik a MSG algoritmus által generált maximális struktúrától, ha legalább egy redundánsan kiterjeszthető entitás típusú csomópontot tartalmaz.*

Tétel 5.3.3 *Az RG algoritmus véges, és lépéseinek száma a maximális struktúrában található redundánsan kiterjeszthető entitások bemenetei számosságának és a bemeneti hatványhalmazai számosságának összege.*

Bizonyítás *Bármely PNS feladat esetén, adott az $\mathcal{M}$ az entitás típusú csomópontok véges számú halmaza, vagyis $|\mathcal{M}| < \infty$. A feladat maximális struktúrája $(\mathfrak{m}, \mathfrak{o})$, amely legfeljebb a feladatban definiált entitásokat tartalmazza, azaz $|\mathfrak{m}| \leq |\mathcal{M}|$. A PNS feladat előzetes analízise nélkül, azaz a redundánsan kiterjeszthető entitások ismerete nélkül felső becslés adható az RG algoritmus futási idejének S_{RG} számosságára, amely felső becslés az 5.3.1 egyenlettel definiálható.*

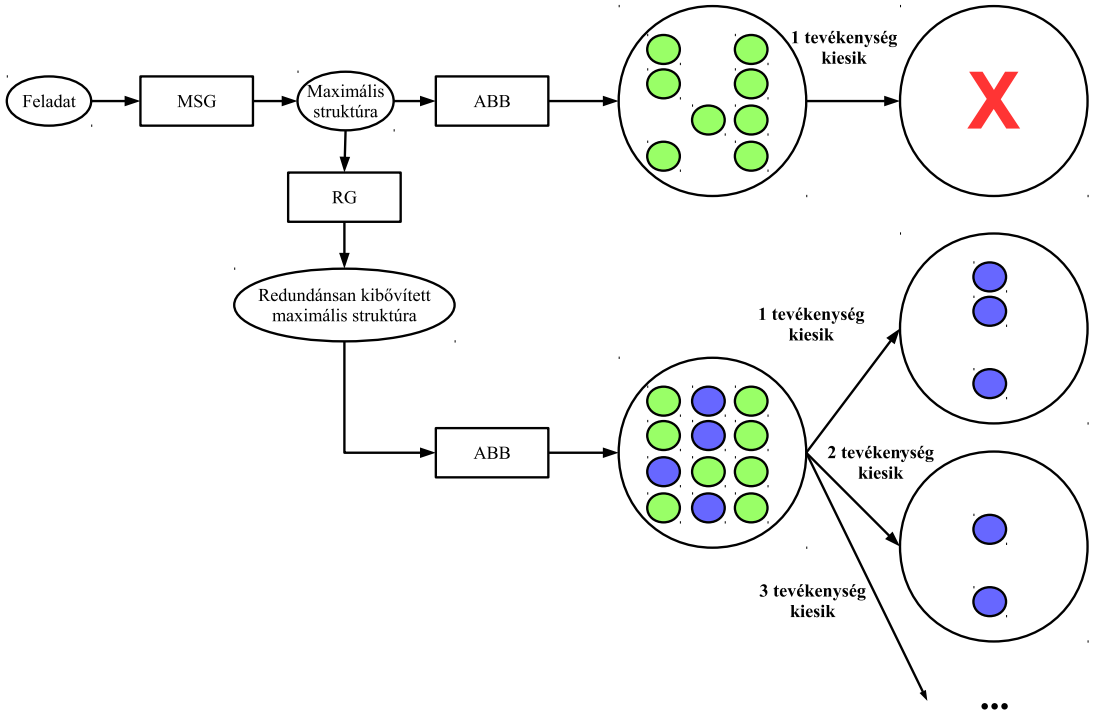
$$S_{RG} \leq \sum_{m_k \in \mathfrak{m}} |\varphi^-(\{m_k\})| + 2^{|\varphi^-(\{m_k\})|} \leq \sum_{m_l \in \mathcal{M}} |\varphi^-(\{m_l\})| + 2^{|\varphi^-(\{m_l\})|} \leq c_1 * 2^{|\mathcal{M}|} \quad (5.3.1)$$

Ahol, $|\varphi^-(\{m_k\})|$ az RG algoritmus 3. lépésében kezdődő ***while*** *ciklus lépésszáma, a $2^{|\varphi^-(\{m_k\})|}$ pedig a 4. lépésben definiált* ***for*** *ciklus lépésszáma és ezek összege pedig minden az 1. lépésben összegyűjtött redundánsan kiterjeszthető entitásra megismétlődik, amely ismétlődést a 2. lépés* ***while*** *ciklusa testesít meg. Továbbá a $c_1 \geq 0$ egy alkalmasan választott állandó és $|\mathcal{M}| \geq c_2$. Azaz az RG $O(2^{|\mathcal{M}|})$ futási idejű.*

Az RG algoritmus lépéseinek a száma a redundánsan kiterjeszthető entitások ismeretében az 5.3.2 egyenlettel definiálható.

$$S_{RG} = \sum_{m_j \in \mathfrak{m} \text{ és } |\varphi^-(\{m_j\})|>1} |\varphi^-(\{m_j\})| + 2^{|\varphi^-(\{m_j\})|} \quad (5.3.2)$$

Az 5.9. ábrán látható diagram szemlélteti az RG algoritmus szerepét. A maximális struktúrából kiindulva az ABB algoritmussal eljuthatunk a feladat optimális megoldás struktúrájához. Ha nem kényszerítjük, akkor az ABB a költség optimalizálás során megszünteti a redundanciákat, így az eredményül kapott hálózatok, már egyetlen



5.9. ábra. Az RG algoritmus szerepének szemléltetése

tevékenység kiesése után is megoldás esetén már egyetlen tevékenység kiesésével is ellentmondáshoz jutnánk. működésképtelenné válnak. Viszont az RG algoritmussal redundánsan kibővített maximális struktúrát alapul véve, és azon az ABB algoritmust futtatva olyan alternatív hálózatokat is kapunk, amelyből egy, kettő vagy akár több tevékenységet elhagyva még mindig megengedett megoldáshoz jutunk.

5.4. Megbízhatóság meghatározása a P-gráf módszertanban

A folyamatok P-gráffal történő leírásakor az entitás jellegű csomópontok bizonyos információkat, minőségi paramétereket, azaz állapotokat tárolnak, a tevékenységek pedig az egyes állapotok közötti változásokat eredményezik. Ezeknek kapcsolatait P-gráffal írjuk le, ahol megadjuk, hogy melyik végcél milyen input információk és tevékenységek esetén működőképes, és az érkező inputok alapján milyen szolgáltatást

nyújt.

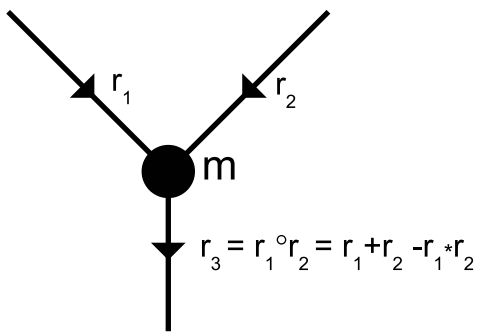
Minden entitás csak egy állapotot rögzíthet. Ha több objektumot azonos szempontból vizsgálunk, akkor redundánsan annyi entitást fog tartalmazni a modellünk, ahány objektumot vizsgálunk.

Minden entitás típusú csúcshoz információ-halmazt rendelhetünk, amely tartalmazhatja az adott állapot megbízhatóságát is. Ez speciális esetben lehet egy valószínűségi érték. Az állapotváltozók megjelennek mind a bemenet, mind a kimenet oldalán. Egy állapotváltozó a megbízhatósági értékét több csomópont megbízhatósági értéke alapján határozza meg. Ilyen esetekben például az alábbi lehetőségek közül választhatunk:

- a valószínűségszámításból ismert eszköztárat használjuk fel a megbízhatósági érték meghatározására, vagy
- a fuzzy-logika által biztosított eszköztárat hívjuk segítségül a megbízhatósági jellemzők megállapításához.

A jelen munkámban az előbbi lehetőséget alkalmazzuk. Következő lépésként a P-gráf egyes állapotaihoz rendelhető minőségi értékeit illetve ennek számítási metódusát kell rögzíteni. A minőségi értékek meghatározásakor két szempontot kell figyelembe vennünk. Ezek az alábbiak:

- Minden tevékenységhez hozzárendelhető egy minőségi érték, és ezen értékek ismeretét feltételezzük továbbiakban (ismertnek feltételezzük például azt, hogy az erőforrások milyen valószínűséggel állnak rendelkezésre).
- Minden entitáshoz megadható egy megbízhatósági, minőséget meghatározó érték. Ezen érték meghatározása a bemenetek számától (és természetesen értékétől) függ a következőképpen:
  - Ha egy entitás jellegű csomópontnak egy bemenő éle van, akkor a kimenő él minősége megegyezik a bemenő él minőségével.



5.10. ábra. Entitás minőségi értékének meghatározása két bemenő él esetén

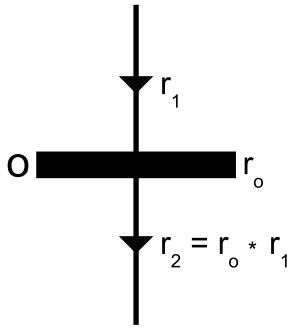
– Több bemenő él esetén a kimenő él minőségére a valószínűségszámításból ismert Poincare-formula használható. Mint ismeretes, ez a formula események uniójának valószínűségére ad meg összefüggést. Ezt általánosítva az adott esetre, az alábbi összefüggés definiálható, amely a bemenő megbízhatósági érték ismeretében „állítja elő" a kimenő él minőségi értékét. Ez az összefüggés két bemenő él esetén az 5.10 ábrán látható módon adható meg:

$$r_1 \circ \cdots \circ r_p = \sum_{i=1}^{p} r_i - \sum_{1 \le i < j \le p} r_i * r_j + \sum_{1 \le i < j < k \le p} r_i * r_j * r_k - \cdots + (-1)^{p-1} * r_1 * \cdots * r_p \quad (5.4.1)$$

Az állapotok minőségi értékeinek definiálása után bemutatjuk azon eseteket, amikor tevékenységen – és nem entitáson – halad át az információ. Az előző esethez hasonlóan itt is két részfeladat definiálható.

- Elsőként tekintsük azt az esetet, amikor egy tevékenységnek – amelynek minőségi értékét jelöljük r_o-val – egy bemeneti éle van. Feltéve, hogy a bemeneti él minőségi értékét r_1-gyel jelöljük, a kimeneti él minőségét e két érték szorzatával, $r_o * r_1$-gyel jellemezhetjük, ahogy az az 5.11 ábrán látható.

- Az előbbiekben ismertetett eset általánosításaként a következőkben bevezetjük a $p \in \aleph$ bemenetű tevékenységek kimenetére alkalmazott összefüggésünket, nevezetesen, ilyen tulajdonságú gráf-csomópont esetén a kimeneti él értékét a bemeneti minőségek és a tevékenység minőségi értékének szorzatával adjuk meg.



5.11. ábra. Tevékenységen áthaladó információ minőségének számítása egy bemenő él esetén

Mindezt formálisan leírva: $r_1, r_2, \ldots, r_p$ bemeneti értékű élek, valamint r_o minőségű tevékenység esetén a kimenő él minősége a következő szorzat:

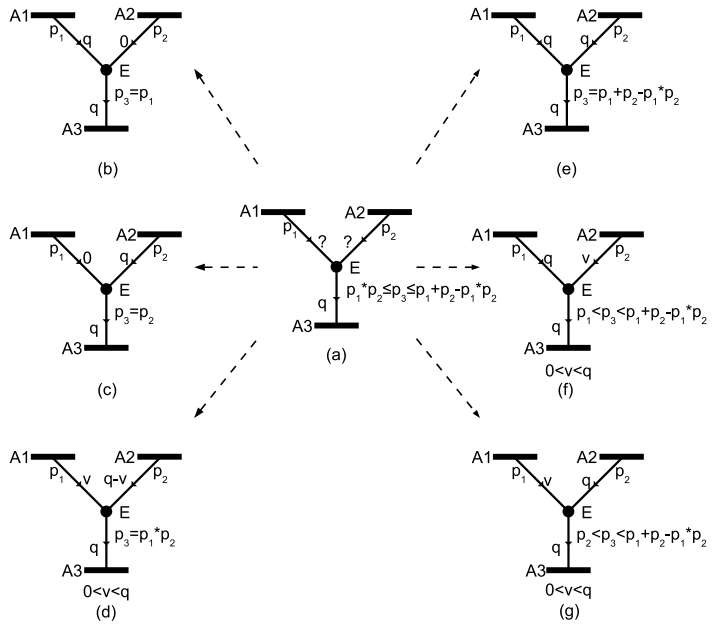
$$r_o \prod_{i=1}^{p} r_i \tag{5.4.2}$$

A most bevezetett 5.4.1 és 5.4.2 összefüggések felhasználásával bármely körmentes és csakis körmentes struktúra megbízhatósága egyszerűen meghatározható úgy, hogy az erőforrásoktól végigkövetjük az entitások és tevékenységek sorozatát a végcélokig, az éppen aktuális formulát alkalmazva ([65], [63], [64]). Irányítatlan kört tartalmazó struktúrák megbízhatóságának meghatározására az 5.7. fejezetben láthatunk pontos számítási módszert.

5.5. A redundancia és a megbízhatóság kapcsolata

Az 5.2. fejezetben bemutatásra került, hogy ha a kapacitás korlátokat is figyelembe kell venni, akkor két tevékenység VAGY kapcsolata esetén hat lehetséges szcenárió különböztethető meg. Az 5.12. ábra szemlélteti az 5.6. ábrán látható különböző eseteket a megbízhatóság szempontjából. A következőkben kiegészítjük a már korábban bemutatott eset leírásokat ([65], [63], [64]).

Az 5.12.a ábra a kiindulási állapot, amelyből aztán hatféle eset állhat elő, amelyek a következőek. Az 5.12.b ábrán egyedül az A1 tevékenység eredményezi az E entitást,



5.12. ábra. Két tevékenység együttműködésének alternatív strukturális reprezentációi és azok megbízhatóságai

azaz az E entitás kimeneti élének p_3 megbízhatósága egyedül A1 tevékenység kimeneti élének p_1 megbízhatóságától függ, vagyis $p_3 = p_1$, az A2 tevékenység pedig nem vesz részt a folyamatban. Az 5.12.c ábrán egyedül az A2 tevékenység eredményezi az E entitást, azaz az E entitás kimeneti élének p_3 megbízhatósága egyedül A2 tevékenység kimeneti élének p_2 megbízhatóságától függ, vagyis $p_3 = p_2$, az A1 tevékenység pedig nem vesz részt a folyamatban. Az 5.12.d ábrán az A1 és az A2 tevékenység együttesen eredményezik az E entitás megkövetelt mennyiségét, azaz az E entitás kimeneti élének p_3 megbízhatósága együttesen az A1 és az A2 tevékenység kimeneti élének p_1 és p_2 megbízhatóságától függ, vagyis $p_3 = p_1 * p_2$. Az 5.12.e ábrán mind az A1 és mind az A2 tevékenység biztosítja a szükséges mennyiséget az E entitásból, azaz redundánsan párhuzamosak, vagyis ebben az esetben a legnagyobb a megbízhatósága az E entitás kimeneti élének, ami $p_3 = p_1 + p_2 - p_1 * p_2$. Az 5.12.f ábrán az A1 tevékenység a teljes q mennyiséget, míg az A2 tevékenység csak egy kisebb v mennyiséget eredményez az E entitásból, vagyis csak részben redundánsak, így az E entitás kimeneti élének p_3 megbízhatósága nagyobb, mint önmagában a p_1 megbízhatóság, de kisebb mintha

teljesen redundánsak lennének, azaz $p_1 < p_3 < p_1 + p_2 - p_1 * p_2$. Végül pedig az 5.12.g ábra azt a szcenáriót szemlélteti amikor az A2 tevékenység a teljes q mennyiséget, míg az A1 tevékenység csak egy kisebb v mennyiséget eredményez az E entitásból, vagyis szintén csak részben redundánsak, így az E entitás kimeneti élének p_3 megbízhatósága nagyobb, mint önmagában a p_2 megbízhatóság, de kisebb mintha teljesen redundánsak lennének, azaz $p_2 < p_3 < p_1 + p_2 - p_1 * p_2$.

Az 5.13 ábra szemlélteti az 5.7 ábrán látható redundánsan kiegészített különböző eseteket a megbízhatóság szempontjából. Az egyes esetek megbízhatóságai megegyeznek az 5.12 ábrán feltüntetett megbízhatóságokkal, viszont a kiegészítés által egyértelművé vált az egyes esetek jelentése.

5.6. Becslési eljárás megbízhatóság számításra

A 5.4. fejezetben bemutatott 5.10. ábrát és 5.4.1. egyenletet felhasználva becslési eljárás adható az entitás jellegű csomópontok kimenő élének (akár többnek is) megbízhatósági korlátaira. Tekintsük a 5.6.1 egyenletet, amely két bemeneti éllel és egy kimeneti éllel rendelkező entitás esetén a kimenti él megbízhatóságát írja le:

$$r_3 = r_1 + r_2 - r_1 * r_2 \tag{5.6.1}$$

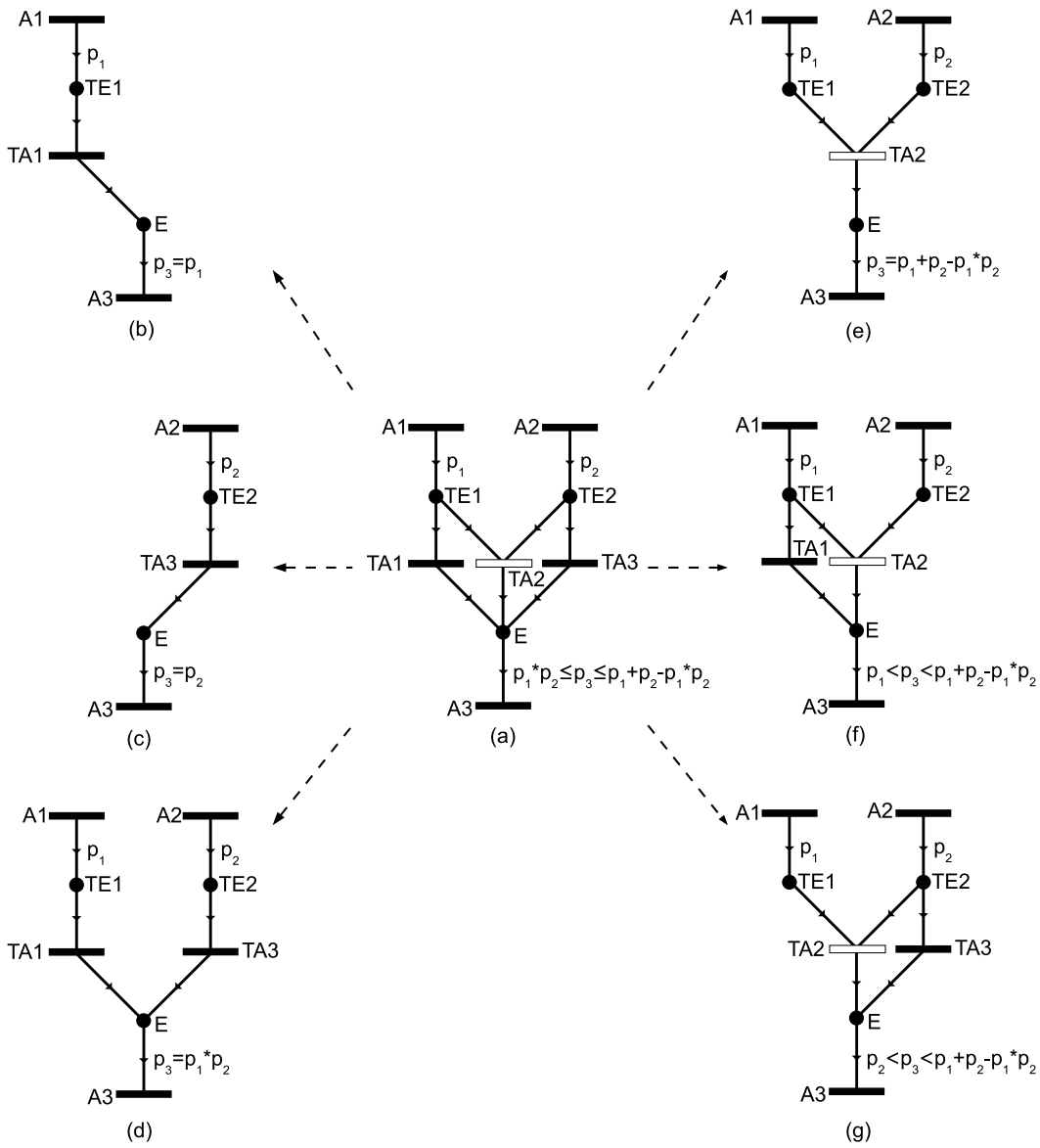
A relaxált modell a következőképpen írható fel:

$$r_1 \leq r_{1max} * y_1, r_2 \leq r_{2max} * y_2 \tag{5.6.2}$$

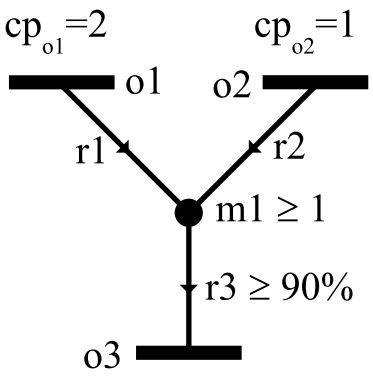
$$r_{3max} = r_{1max} + r_{2max} - r_{1max} * r_{2max} \tag{5.6.3}$$

$$r_3 \leq r_{3max} \tag{5.6.4}$$

A 5.6.2. egyenlet az egyes bementi élek megbízhatósági értékeire ad felső korlátot, mégpedig úgy, hogy vesszük az egyes élek megbízhatósági értékeinek a maximumát. Az y_i (i=1,2,..) értékek pedig azt határozzák meg, hogy az egyes élekhez tartozó tevékenységek – amelyekből az élek kimenetként kilépnek és az adott entitás bemeneteként jelennek meg – szerepelnek-e a hálózatban ($y_i = 1$) vagy sem ($y_i = 0$). Majd a



5.13. ábra. Két párhuzamos tevékenység alternatív szcenárióinak szuperstruktúrája



5.14. ábra. A megbízhatósági korlátra adott becslési eljárást szemléltető példa

bemeneti élek megbízhatóságainak felső korlátjait felhasználva kiszámolható a kimeneti élnek (akár többnek is) a megbízhatósági értéke a 5.6.3. egyenlet szerint, amely egyben egy felső korlátot is ad az entitás kimeneti élének megbízhatóságára, ahogy az a 5.6.4. egyenleten látható.

Ezzel a becslési eljárással kiegészítve az 5.4. fejezetben bemutatott modellt, már figyelembe tudjuk venni a különböző megbízhatósági korlátokat. A 5.14. ábrán látható példán mutatom be a becslési eljárás működését.

A feladatban 3 tevékenység (o1, o2, o3) és egy entitás jellegű csomópont (m1) szerepel. Ahogy az a 5.14. ábrán látható az m1 entitásra bevezetünk egy alsó korlátot, amely annyit jelent, hogy legalább 1 egységnyi termelni kell belőle, illetve az r_3 megbízhatóságra, hogy legalább 90%-os valószínűséggel elő kell állnia az m1 entitásnak. A feladat a további paraméterekkel és célfüggvénnyel rendelkezik:
$r_{1max} = 95\%, r_{2max} = 90\%$
$r_3 \leq r_{3max} = 95\% + 90\% - 95\% * 90\% = 0,995\%$
$2 * x1 + x2 \longrightarrow min$

A 5.1. táblázat foglalja össze a különböző megbízhatósági szintek megkövetelése esetén kapott eredményeket. Négy esetet ír le, amelyek közül az első eset az amikor a feladat által megkövetelt minimális 90%-os megbízhatóságot kívánjuk teljesíteni. Ebben az esetben csak az o2 tevékenységre van szükségünk ahhoz, hogy a megfelelő megbízhatóság mellett előálljon az m3 entitás. A második eset az amikor 95%-os,

5.1. táblázat. A megbízhatósági korlátra adott becslési eljárást szemléltető példa eredményei

Megbízhatóság	x1	x2	Becsült költség	Valós költség
90%	0	1	1	1
95%	0,053	1	1,1	2
99%	0,095	1	1,189	3
100%	Nem eredményez megoldást			

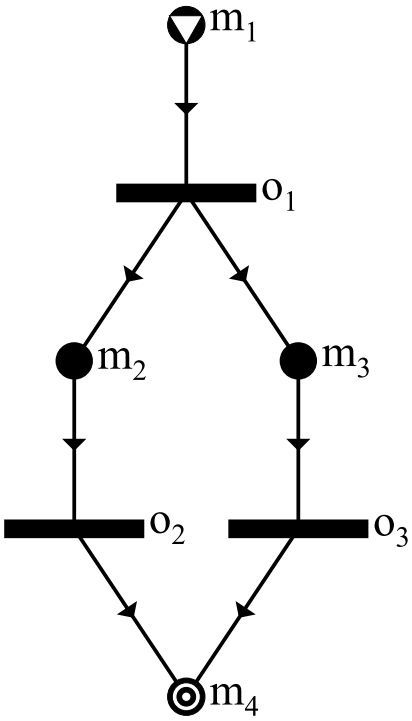
a harmadik pedig amikor 99%-os megbízhatósági szintet követelünk meg. Ahogy azt korábban láthattuk az r_3 maximálisan 0,995%- os valószínűségű lehet, ezért a negyedik eset amikor 100% próbálunk előírni nem eredményezhet megoldást.

A fenti korlátozási feltétel hasonlóan írható fel tetszőleges hálózat elemeire. Ezen korlátokat az optimális hálózat keresése során használt relaxált matematikai programozási modellbe építve két szempontból is javíthatjuk a keresés hatékonyságát. Egyrészt a korlátok eredményeként – ahogy a példából is látható – nagyobb megkövetelt megbízhatósággal nő a relaxált feladat költsége, közelítve a valós költséget. Másrészt, bizonyos tevékenységek kiesése esetén, ha a megbízhatósági korlát már nem teljesíthető, a modell megoldhatatlanná válik, és a keresés ebbe az irányba nem folytatódik.

5.7. Tetszőleges struktúrák megbízhatóságának meghatározása

A redundancia generálás a hálózat struktúrájától függetlenül javítja a megbízhatóságot, még ha annak értékét általában nem is olyan könnyű kiszámolni és szintézis során figyelembe venni, mint ahogy azt, az előző fejezetekben bemutattuk. A most következő fejezetben ismertetjük, hogy miként számítható ki tetszőleges struktúrájú hálózat megbízhatósága. Erre Kovács és szerző társai adtak módszert, amely a Bayes tétel alapján teljes leszámlálással határozza meg a hálózat megbízhatósági értékét ([67]).

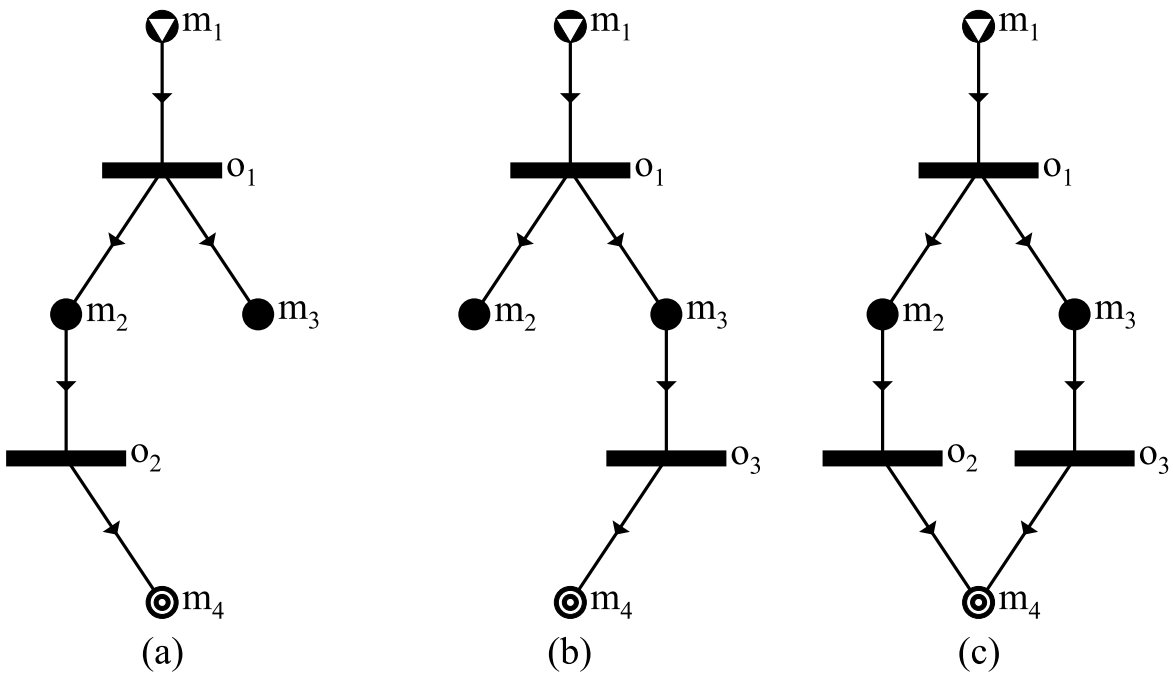
Lemma 5.7.1 *A redundancia generálás a kört tartalmazó struktúrák megbízhatóságát is növeli, csak a megbízhatósági értéket nem az 5.4.1 és 5.4.2 összefüggések alapján kell meghatározni, mint ahogy az az előzőekben bemutatásra került, hanem a Kovács és szerzőtársai által javasol Bayes tétel alapú módszerrel.*



5.15. ábra. Az irányítatlan kört tartalmazó szemléltető példa maximális struktúrája

Az 5.7.1. lemma szemléltetésére tekintsük az 5.15. ábrán látható példát, amelyben három tevékenység az o_1, az o_2 és az o_3 szerepelnek, továbbá az m_1 erőforrás, az m_2 és az m_3 köztes entitások és az m_4 végcél. Irányítatlan kör akkor állhat elő, ha elhagyjuk a feladatot ábrázoló P-gráf éleinek irányítását.

A feladat megoldása során kombinatorikusan négy struktúra állhat elő. Az első eset, amikor nem választunk be egy tevékenységet sem a struktúrába, de mivel jelen feladat szempontjából ennek nincs jelentősége, ezért a továbbiakban ezzel nem is foglalkozunk. A második eset amikor az, o_1 és az o_2 tevékenységek kerülnek a struktúrába. A harmadik eset az, amikor az o_1 és az o_3 tevékenységek kerülnek a struktúrába.



5.16. ábra. Az irányítatlan kört tartalmazó szemléltető példa lehetséges megoldás struktúrái

A negyedik pedig az, amikor az o_1, az o_2 és az o_3 tevékenységek mindegyike szerepel a struktúrában. Az imént bemutatott második, harmadik és negyedik esetet az 5.16 ábra szemlélteti, amelyek megbízhatóságait – a Kovács és szerző társai által megadott módszer alapján – az 5.2 táblázat szemlélteti.

A módszer leszámlálással végigveszi és összegzi az egyes esetek összes olyan lehetőséges részhálózatának megbízhatóságát, amely eredményre vezethet. Továbbá ha az adott részhálózatban szerepel egy, az adott esetbeli tevékenység, akkor azt a megbízhatóságával, míg ha nem szerepel, akkor azt az (1 - megbízhatósága) értékkel vesszük és összeszorozzuk az egy részstruktúrához tartozó megbízhatóság értékeket.

Tehát az alap feladatban az 5.16.a ábrán látható esetben csak egy lehetséges részhálózat állhat elő, mégpedig amelyik mind az o_1 és mind az o_2 tevékenységeket tartalmazza. Az 5.16.b ábrán látható esetben szintén csak egy lehetséges részhálózat állhat elő, mégpedig amelyik mind az o_1 és mind az o_3 tevékenységeket tartalmazza. Végül pedig az 5.16.c ábrán látható esetben szintén csak egy lehetséges részhálózat állhat

elő, mégpedig amelyik mind a három (o_1, o_2, o_3) tevékenységet tartalmazza, mivel az o_2 és az o_3 tevékenységek együttesen eredményezik az m_4 entitást. Az a vagy b esetek valamelyike az imént bemutatott három eset közül a legnagyobb megbízhatóságú struktúra, mivel minden megbízhatósági érték 0 és 1 közé esik, aszerint hogy hány %-osan megbízható az adott tevékenység.

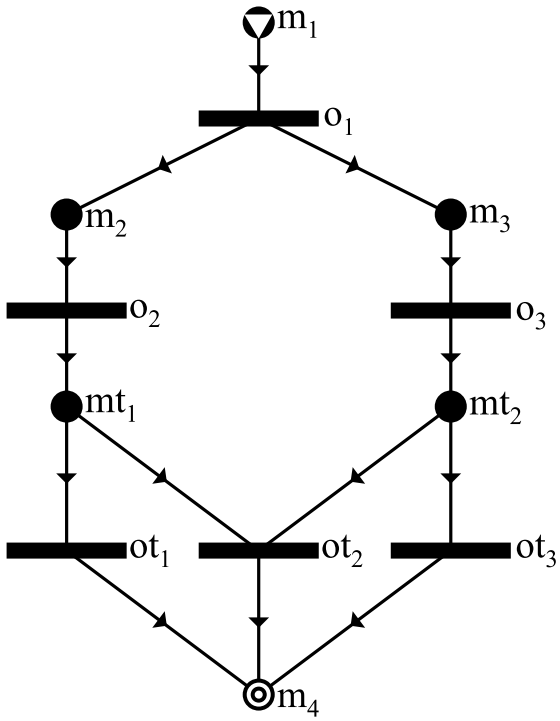
5.2. táblázat. Tetszőleges struktúra megbízhatóságának kiszámítása

Eset	#	**o1**	**o2**	**o3**	**Megbízhatóság**
a	a.1	1	1	-	$p_1 * p_2$
	$p_a = p_{a.1} = p_1 * p_2$				
b	b.1	1	-	1	$p_1 * p_3$
	$p_b = p_{b.1} = p_1 * p_3$				
c	c.1	1	1	1	$p_1 * p_2 * p_3$
	$p_c = p_{c.1} = p_1 * p_2 * p_3$				

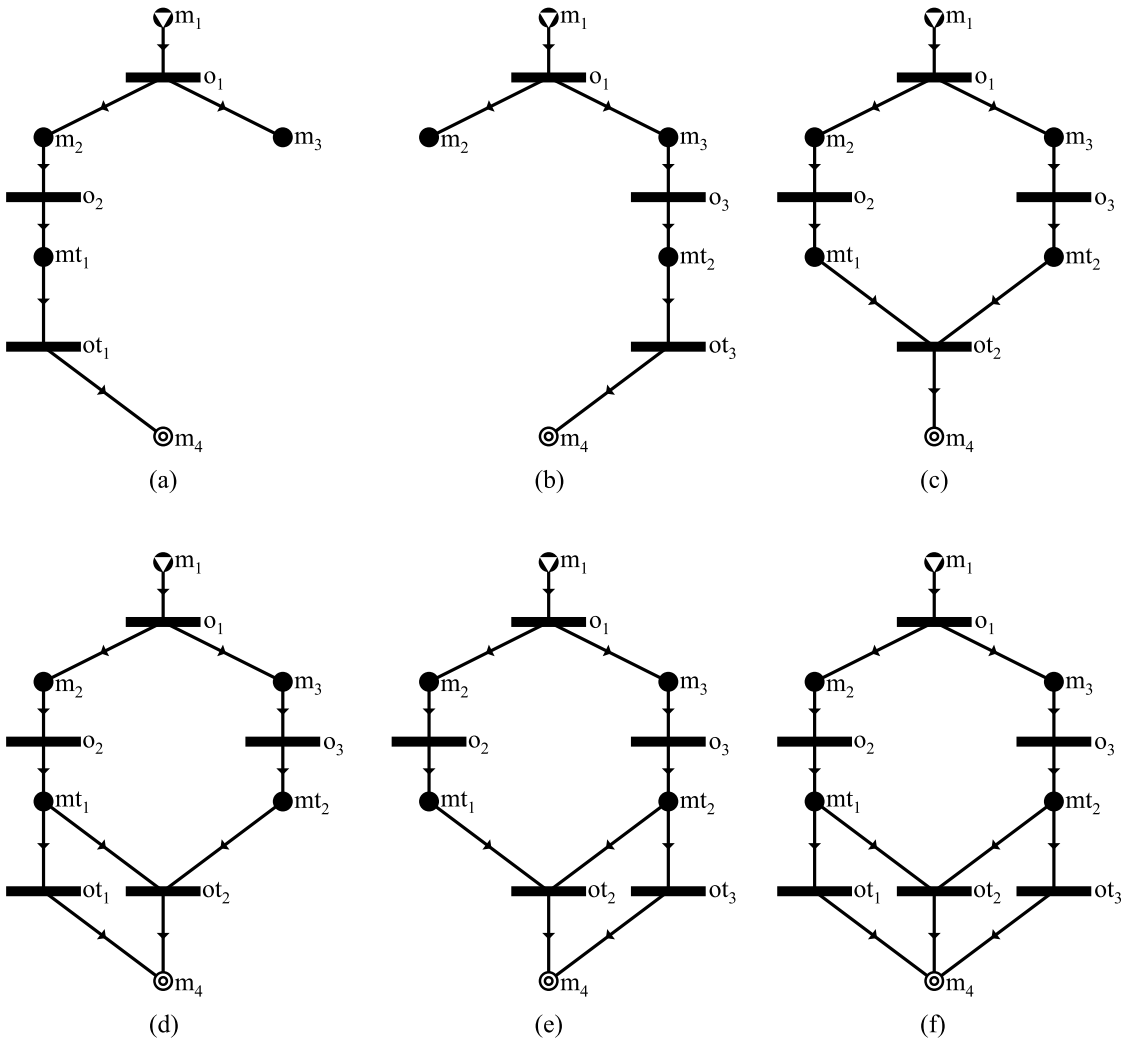
Az 5.7.1. lemmát a példa hálózat redundáns kiegészítésével szemléltetjük, amely a 5.17 ábrán látható. Az eredeti példa hálózat három kiegészítő tevékenységgel ($ot1$, $ot2$ és $ot3$) és az integrálásukhoz szükséges két kiegészítő entitással ($mt1$ és $mt2$) bővült.

A redundánsan kibővített maximális struktúrából kombinatorikusan hat lehetséges struktúra állhat elő, amelyek az 5.18. ábrán láthatóak. Ezen struktúrák megbízhatóságainak kiszámítási módját és értékét az 5.3. táblázat mutatja be. A c, d, e és f esetekben állhat elő több részhálózat is ami eredményre vezethet, ahogy az az 5.3. táblázatban látható. A legnagyobb megbízhatósága az 5.18.c ábrán látható struktúrának van. Ez az amely a teljesen redundáns esetet reprezentálja. Ennek a $p_c = p_1 * p_2 + p_1 * p_3 - p_1 * p_2 * p_3$ megbízhatósága jóval nagyobb mint az alap feladat bármely struktúrájának megbízhatósága, hiszen $p_1 * p_2 + p_1 * p_3 - p_1 * p_2 * p_3 > p_1 * p_2$, $p_1 * p_2 + p_1 * p_3 - p_1 * p_2 * p_3 > p_1 * p_3$ és $p_1 * p_2 + p_1 * p_3 - p_1 * p_2 * p_3 > p_1 * p_2 * p_3$.

Tehát az irányítatlan kört tartalmazó struktúrák megbízhatósága is növelhető redundáns átalakításukkal. Továbbá megbízhatósági értékük számítására Kovács és szerzőtársai adtak módszert.



5.17. ábra. Az irányítatlan kört tartalmazó szemléltető példa redundánsan kiegészített maximális struktúrája



5.18. ábra. A redundánsan kiegészített irányítatlan kört tartalmazó szemléltető példa lehetséges megoldás struktúrái

5.8. Szemléltető példa

Egy az olajiparból származó ellátási-lánc feladaton keresztül szemléltetjük az előzőekben bemutatott új módszert. A megoldandó feladatban egy olajipari vállalat három telephelye szerepel, amelyek Pécs, Dombóvár és Kaposvár. A cél hogy biztosítva legyen a megkövetelt biodízel mennyiség Kaposváron a másik kettő telephelyről minimális összköltséggel és kockázattal.

Megszabott mennyiségű biodízel és komponensei állnak rendelkezésre Dombóváron és Pécsen. A pécsi telephely rendelkezik egy limitált keverési kapacitással a rendelkezésre álló komponensekből történő biodízel előállítására, vagyis ott a négy fő komponens – amelyek a K2 komponens (HDS gázolaj), a K4 komponens (kerozin), a K7 komponens (szulfur nélküli gázolaj) és a K8 komponens(FAME biokomponens) – feldolgozható és abból biodízel állítható elő. A biodízel mindegyik telephelyen tartályokba tölthető, amely tartályok teherautókkal szállíthatóak Kaposvárra, ahol aztán azok lefejthetőek. Mindegyik telephelyen van egy felső korlátja az ott levő erőforrásoknak.

A szemléltető feladat kombinatorikus komponenseit egy hármassal $(\mathcal{P}, \mathcal{R}, \mathcal{O})$ adhatjuk meg, ahogy azt a 2.2.3 fejezetben láthattuk, ahol

$\mathcal{P} = \{$*Biodízel lefejtve a Kaposvári telephelyen*$\}$

$\mathcal{R} = \{$*Biodízel a dombóvári telephelyen*, *Biodízel a pécsi telephelyen*, *K2 komponens*, *K4 komponens*, *K7 komponens*, *K8 komponens*$\}$

$\mathcal{O} = \{$*B, U1, U2, U3, T1, T2, D*$\} =$

$\{(\{$*K2 komponens, K4 komponens, K7 komponens, K8 komponens*$\}$,

$\{$*Biodízel előállítva Pécsen*$\})$,

$(\{$*Biodízel a dombóvári telephelyen*$\}, \{$*Biodízel tartályba töltve Dombóváron*$\})$,

$(\{$*Biodízel a pécsi telephelyen*$\}, \{$*Biodízel tartályba töltve Pécsen*$\})$,

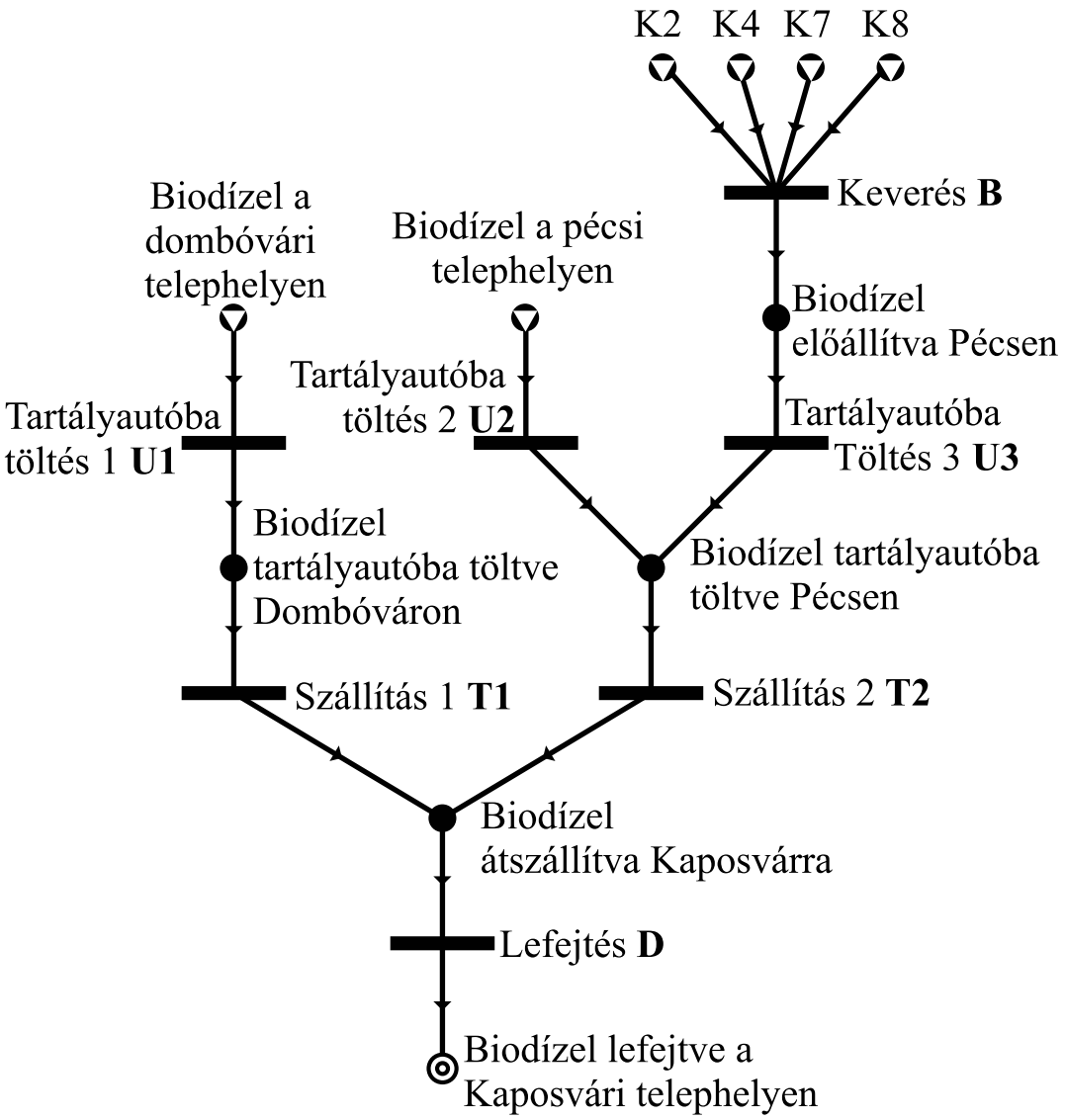
$(\{$*Biodízel előállítva Pécsen*$\}, \{$*Biodízel tartályba töltve Pécsen*$\})$,

$(\{$*Biodízel tartályba töltve Dombóváron*$\}, \{$*Biodízel átszállítva Kaposvárra*$\})$,

$(\{$*Biodízel tartályba töltve Pécsen*$\}, \{$*Biodízel átszállítva Kaposvárra*$\})$,

$(\{$*Biodízel átszállítva Kaposvárra*$\}, \{$*Biodízel lefejtve a Kaposvári telephelyen*$\})\}$

A szemléltető példa tevékenységeit és azok, az előzőekben felsorolt, előfeltételeit és következményeit, illetve a megbízhatóságaikat az 5.4. táblázat foglalja össze.



5.19. ábra. A szemléltető példa maximális struktúrájának P-gráf reprezentációja

A jelenlegi példában két entitás típusú csomóponthoz tartozik több, mint egy bemeneti él, azaz név szerint mind a „Biodízel átszállítva Kaposvárra” és mind a „Biodízel tartályautóba töltve Pécsen” entitás biztosítható két-két tevékenység által. Az RG algoritmus futása után a maximális struktúra kiegészül a TE1, TE2, TE3 és a

TE4 entitásokkal és a TA1, TA2, TA3, TA4, TA5 és a TA6 tevékenységekkel. Az 5.19. ábrán látható az RG algoritmus bemenetéül szolgáló kezdeti maximális struktúra és az 5.20. ábra pedig az RG által kiterjesztett maximális struktúrát szemlélteti. A redundanciával kiterjesztett maximális struktúra olyan lépéseket tartalmaz, amelyek garantálják a redundanciát az ellátási láncban és ennek következtében növelik annak megbízhatóságát. A tevékenységeknek megbízhatósága $97 - 99\%$, ahogy az az 5.4. táblázatban szerepel. Az ellátási szcenáriók megbízhatósága redundancia nélkül a $86,80\%$ és $95,08\%$ közötti tartományban található, míg redundanciával a $93,09\%$ és $98,92\%$ közé esik.

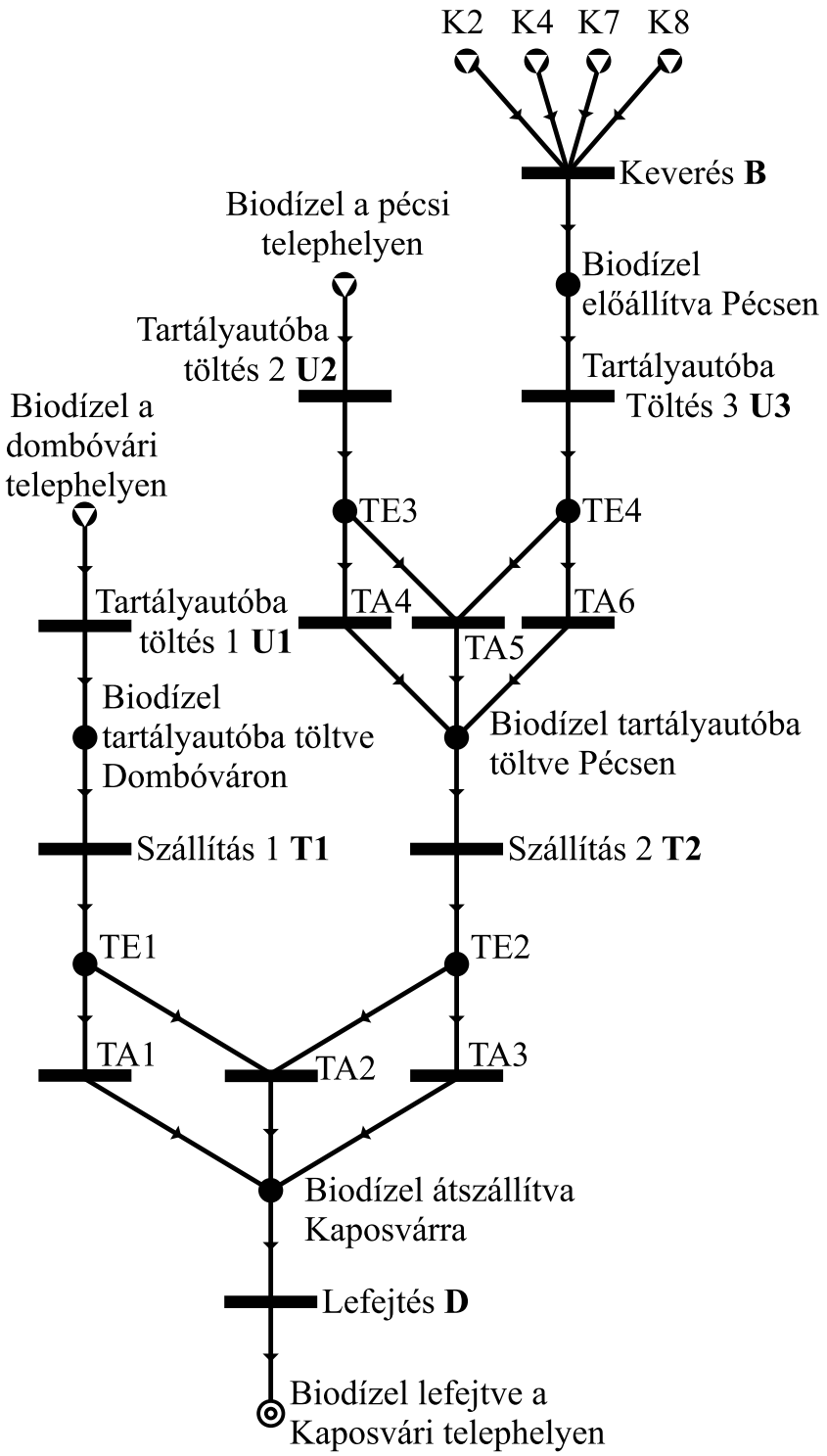
Az SSG algoritmus összesen 43 alternatív megoldás struktúrát eredményez a redundánsan kiterjesztett maximális struktúrából, amelyek mindegyike kombinatorikusan lehetséges. Közülük hét egyáltalán nem tartalmaz redundanciát, hat pedig tartalmaz teljesen redundáns részstruktúrát, vagyis a párhuzamos esetek legalább egyikében a megkövetelt kimenetek garantáltan duplikált mennyiségben állnak rendelkezésre.

A kombinatorikusan lehetséges alternatív struktúrák SSG algoritmussal történő generálása után, a teljes megbízhatóság kiszámolható az előzőekben megadott formulával. Az 5.5. és 5.6. táblázatok szemléltetik a 13 kiválasztott alternatív ellátási szcenárió teljes megbízhatóságait és azok kiszámítási formuláit. A számítási eredmények is jól szemléltetik, hogy a P-gráf módszertan előzőekben bemutatott kiterjesztése segítségével olyan redundáns struktúrák generálhatóak, amelyek magasabb megbízhatósági értékeket eredményeznek. Az 5.21. ábrán látható a legmagasabb megbízhatósági értékkel rendelkező megoldás struktúra P-gráf reprezentációja.

Megjegyzés 5.8.1 *A redundanciával kiterjesztett szuperstruktúrát alapul véve az összes szcenárió generálható az eredeti P-gráf algoritmusokkal, például a PNS Studio szoftverbe integrált SSG algoritmus futtatásával.*

5.9. Összefoglalás

Ismertettem a redundanciában rejlő potenciális lehetőségeket és előnyöket. Bevezettem a megbízhatóság kiszámítási módszerét a P-gráf módszertanban. Algoritmikus



5.20. ábra. A szemléltető példa maximális struktúrájának redundánsan kibővített P-gráf reprezentációja

módszert dolgoztam ki az alap szuperstruktúra kiterjesztésére, amely garantálja a redundanciát és ezáltal egy ellátási szcenáriónak a teljes megbízhatóságát növeli. A redundánsan kiterjesztett maximális struktúrát alapul véve az összes szcenárió generálható az eredeti P-gráf algoritmusokkal.

5.10. Kapcsolódó publikációim

Referált nemzetközi folyóirat cikkek

1. Bertok, B., Kalauz, K., Sule, Z., and Friedler, F., Combinatorial Algorithm for Synthesizing Redundant Structures to Increase Reliability of Supply Chains: Application to Biodisel Suplly, *Industrial & Engineering Chemistry Research*, 52(1):181–186, 2013.

Nemzetközi konferencia előadások

2. Kalauz, K., Sule, Z., Bertok, B., Friedler, F., and Fan L. T., Generation of Redundant Structures to Guarantee Predefined Level of Reliability in Business Processes, VOCAL 2010, Veszprém, Hungary, December 13-15, 2010.

1. Kalauz, K., Sule, Z., Bertok, B., Friedler, F., and Fan L. T., Reliability Analysis of Business Processes via the P-graph Framework, International Conference on Computational Management Science, Vienna, Austria, July 28-30, 2010.

Hazai konferencia előadások

1. Kalauz, K., Sule, Z., Tarczali, T., Bertok, B., Friedler, F., and Fan L. T., P-gráf módszertan alkalmazása biztonságos üzleti folyamatok modellezésére, XXVIII: Magyar Operációkutatási Konferencia, Balatonőszöd, 2009.

5.3. táblázat. A redundánsan kibővített struktúra megbízhatóságának kiszámítása

Eset	#	o_1	o_2	o_3	ot_1	ot_2	ot_3	**Megbízhatóság**
a	a.1	1	1	-	1	-	-	$p_1 * p_2 * 1$
	$p_a = p_{a.1} = p_1 * p_2$							
b	b.1	1	-	1	-	-	1	$p_1 * p_3 * 1$
	$p_b = p_{b.1} = p_1 * p_3$							
c	c.1	1	1	0	-	1	-	$p_1 * p_2 * (1 - p_3) * 1$
	c.2	1	0	1	-	1	-	$p_1 * (1 - p_2) * p_3 * 1$
	c.3	1	1	1	-	1	-	$p_1 * p_2 * p_3 * 1$
	$p_c = p_{c.1} + p_{c.2} + p_{c.3} = p_1 * p_2 + p_1 * p_3 - p_1 * p_2 * p_3$							
d	d.1	1	1	0	0	1	-	$p_1 * p_2 * (1 - p_3) * 0 * 1$
	d.2	1	0	1	0	1	-	$p_1 * (1 - p_2) * p_3 * 0 * 1$
	d.3	1	1	1	0	1	-	$p_1 * p_2 * p_3 * 0 * 1$
	d.4	1	1	1	1	1	-	$p_1 * p_2 * p_3 * 1 * 1$
	$p_d = p_{d.1} + p_{d.2} + p_{d.3} + p_{d.4} = p_1 * p_2 * p_3$							
e	e.1	1	1	0	-	1	0	$p_1 * p_2 * (1 - p_3) * 1 * 0$
	e.2	1	0	1	-	1	0	$p_1 * (1 - p_2) * p_3 * 1 * 0$
	e.3	1	1	1	-	1	0	$p_1 * p_2 * p_3 * 1 * 0$
	e.4	1	1	1	-	1	1	$p_1 * p_2 * p_3 * 1 * 1$
	$p_e = p_{e.1} + p_{e.2} + p_{e.3} + p_{e.4} = p_1 * p_2 * p_3$							
f	f.1	1	1	0	0	1	0	$p_1 * p_2 * (1 - p_3) * 0 * 1 * 0$
	f.2	1	0	1	0	1	0	$p_1 * (1 - p_2) * p_3 * 0 * 1 * 0$
	f.3	1	1	1	0	1	0	$p_1 * p_2 * p_3 * 0 * 1 * 0$
	f.4	1	1	1	1	0	1	$p_1 * p_2 * p_3 * 0 * 1 * 1$
	f.5	1	1	1	1	1	1	$p_1 * p_2 * p_3 * 0 * 1 * 1$
	$p_f = p_{f.1} + p_{f.2} + p_{f.3} + p_{f.4} + p_{f.5} = p_1 * p_2 * p_3$							

5.4. táblázat. A tevékenységek előfeltételei és következményei

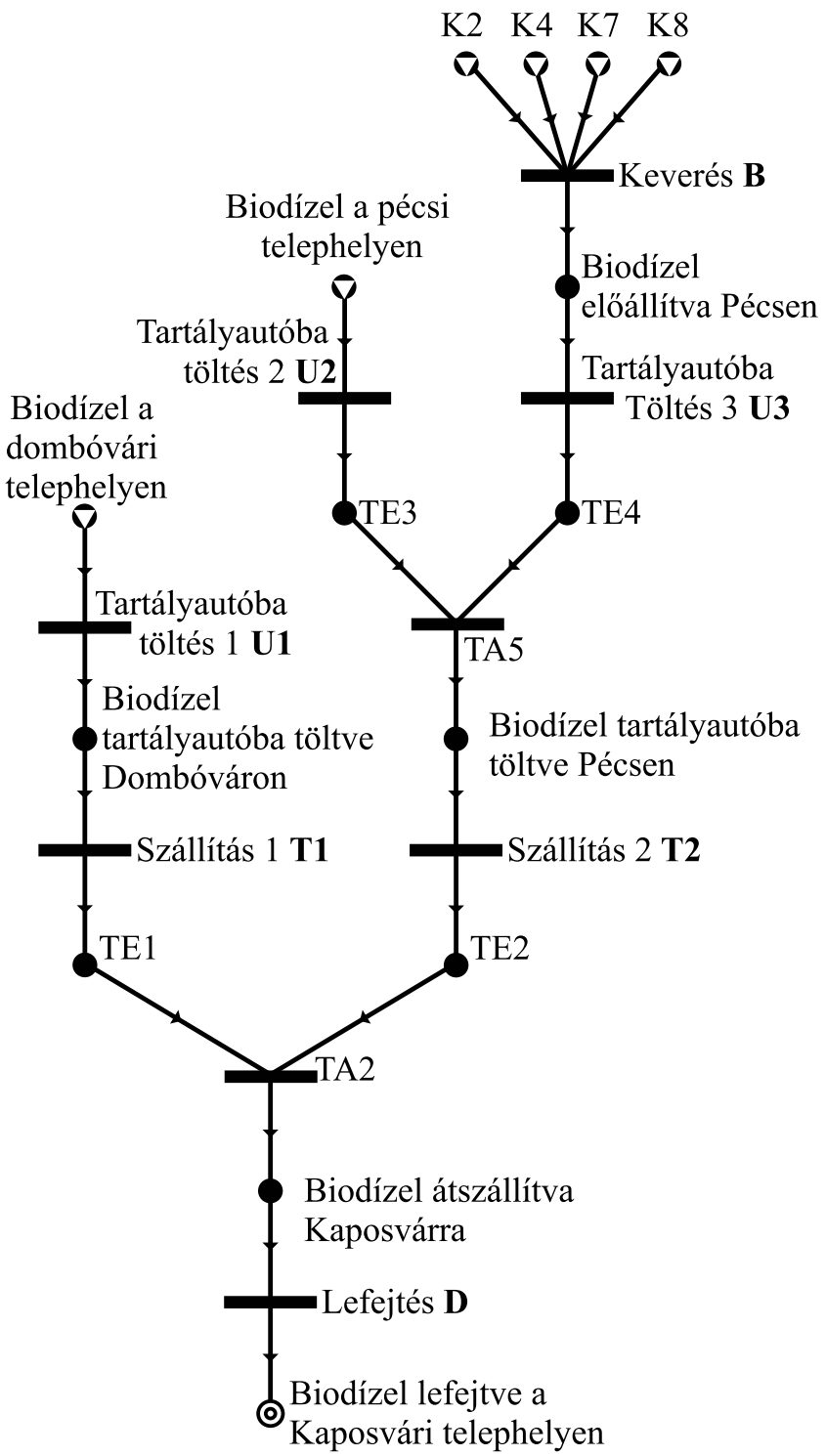
Tevékenységek	**Előfeltételek**	**Következmények**	**Megbízhatóság**
Keverés (B)	K2 komponens, K4 komponens, K7 komponens, K8 komponens	Biodízel előállítva Pécsen	$r_B = 98\%$
Tartályautóba töltés 1 (U1)	Biodízel a dombóvári telephelyen	Biodízel tartályautóba töltve Dombóváron	$r_{U1} = 98\%$
Tartályautóba töltés 2 (U2)	Biodízel a pécsi telephelyen	Biodízel tartályautóba töltve Pécsen	$r_{U2} = 98\%$
Tartályautóba töltés 3 (U3)	Biodízel előállítva Pécsen	Biodízel tartályautóba töltve Pécsen	$r_{U3} = 98\%$
Szállítás 1 (T1)	Biodízel tartályautóba töltve Dombóváron	Biodízel átszállítva Kaposvárra	$r_{T1} = 98\%$
Szállítás 2 (T2)	Biodízel tartályautóba töltve Pécsen	Biodízel átszállítva Kaposvárra	$r_{T2} = 98\%$
Lefejtés (D)	Biodízel átszállítva Kaposvárra	Biodízel lefejtve a Kaposvári telephelyen	$r_D = 98\%$

5.5. táblázat. Az alternatív ellátási szcenáriók teljes megbízhatósága

Struktúra	**Tevékenységek**	**Teljes megbízhatóság**	**Redundancia (igen/nem)**
Str1	D, TA1, T1, U1	95,08%	nem
Str2	D, TA2, T1, U1, T2, TA4, U2	98,84%	igen
Str3	D, TA2, T1, U1, T2, TA5, U2, U3, B	98,92%	igen
Str4	D, TA2, T1, U1, T2, TA5, U3, B	98,73%	igen
Str5	D, TA2, T1, U1, T2, TA4, U2, TA6, U3, B	98,66%	igen
Str6	D, TA3, T2, TA4, U2	95.08%	nem
Str7	D, TA3, T2, TA5, U2, U3, B	96,92%	igen
Str8	D, TA3, T2, TA6, U3, B	92,23%	nem
Str9	D, TA3, T2, TA4, U2, TA6, U3, B	90,38%	nem
Str10	D, TA1, T1, U1, TA3, T2, TA4, U2	91,31%	nem
Str11	D, TA1, T1, U1, TA3, TA5, U2, U3, B	93,09%	igen
Str12	D, TA1, T1, U1, TA3, T2, TA6, U3, B	88,58%	nem
Str13	D, TA1, T1, U1, TA3, T2, TA4, U2, TA6, U3, B	86,80%	nem

5.6. táblázat. Az alternatív ellátási szcenáriók teljes megbízhatóságának kiszámítási formulái

Struktúra	**Teljes megbízhatóság**
Str1	$r_{U1} * r_{T1} * r_D = 95.08\%$
Str2	$(r_{U1} * r_{T1} + r_{U2} * r_{T2} - r_{U1} * r_{T1} * r_{U2} * r_{T2}) * r_D = 98.84\%$
Str3	$\{r_{U1} * r_{T1} + (r_{U2} + r_B * r_{U3} - r_{U2} * r_B * r_{U3}) * r_{T2} - r_{U1} * r_{T1} * [(r_{U2} + r_B * r_{U3} - r_{U2} * r_B * r_{U3}) * r_{T2}]\} * r_D = 98.92\%$
Str4	$(r_{U1} * r_{T1} + r_B * r_{U3} * r_{T2} - r_{U1} * r_{T1} * r_B * r_{U3} * r_{T2}) * r_D = 98.73\%$
Str5	$(r_{U1} * r_{T1} + r_B * r_{U3} * r_{U2} * r_{T2} - r_{U1} * r_{T1} * r_B * r_{U3} * r_{U2} * r_{T2}) * r_D = 98.66\%$
Str6	$r_{U2} * r_{T2} * r_D = 95.08\%$
Str7	$(r_{U2} + r_B * r_{U3} - r_{U2} * r_B * r_{U3}) * r_{T2} * r_D = 96.92\%$
Str8	$r_B * r_{U3} * r_{T2} * r_D = 92.23\%$
Str9	$r_B * r_{U3} * r_{U2} * r_{T2} * r_D = 90.38\%$
Str10	$r_{U2} * r_{T2} * r_{U1} * r_{T1} * r_D = 91.31\%$
Str11	$[((r_{U2} + r_B * r_{U3} - r_{U2} * r_B * r_{U3} * r_{T2}) * r_{U1} * r_{T1})] * r_D = 93.09\%$
Str12	$r_B * r_{U3} * r_{T2} * r_{U1} * r_{T1} * r_D = 88.58\%$
Str13	$r_B * r_{U3} * r_{U2} * r_{T2} * r_{U1} * r_{T1} * r_D = 86.80\%$



5.21. ábra. A legmagasabb megbízhatósággal rendelkező megoldás struktúra P-gráf reprezentációja

6. fejezet

Időkorlátos PNS (TCPNS)

Napjainkban egyre rövidebb idő alatt egyre nagyobb változások következnek be, a népesség egyre rövidebb idő alatt duplázódik, fokozott ütemben nő a bennünket elérő információmennyiség, és egyre rövidebb idő alatt egyre nagyobb távolságokat tudunk elérni. Mindez azt a szubjektív érzetet keltheti, hogy gyorsul a világ, amiben élünk, és az idő szűkös erőforrássá válik, amiből egységnyi mennyiséget egyre több dologra kellene hasznosítanunk([103]). Az idő szemlélet változása nem újkeletű, nem köthető az információs társadalom kialakulásához. Az idő értékének fölismerése az ipari társadalmakhoz kapcsolódik, ahol az nem céltalanul és nem a természet diktálta körforgásban zajlott, hanem anyagi értékek forrásává vált. Az utóbbi néhány évtizedben kialakult, és növekszik egy olyan időérzékeny réteg a társadalomban, amely számára az idő, mint pótolhatatlan erőforrás egyre értékesebbé válik, fogyasztóként pedig az a legfőbb érdeke, hogy minél kevesebb idő áldozat árán jusson hozzá a szükségesnek ítélt dolgokhoz. A hozzájutás ideje az ellátási lánc magasabb szintjein elhelyezkedő többi szereplő számára még fontosabb kérdés, mivel meghatározza saját belső működésüket, de vevőik kiszolgálásán keresztül külső sikerességükre is alapvető hatással van. A termékek idő és helyértéke így megnő, aminek következményeként az egyedi ellátók versenyét az ellátási láncok, ellátási hálózatok versenye váltja föl. A láncok erősségét leggyengébb láncszemük határozza meg, így minden szereplőnek érdeke, hogy mind a belső idő menedzsment, mind a külső kapcsolatok kezelése biztosítsa a gyors és rugalmas működést. Az ellátási folyamat időszükségletének legnagyobb

hányada helyváltoztatással és várakozással telik, a tényleges előállítás csak elenyésző időt vesz igénybe. Így kézenfekvő a logisztikában keresni az idő rövidítés lehetőségeit. Az időszükséglet alakulásának különös a jelentősége a globális termelés - beszerzés - ellátás világában, ezek pedig térben egyre hosszabb ellátási láncokon zajlanak.

A most következő fejezetben bemutatjuk az időkorlátok kezelésével kibővített PNS feladatot és módszert adok a szinkron és aszinkron működésű megoldások generálására. Mint azt már korábban láthattuk, folyamathálózatok szintézisekor adott számú legoptimálisabb folyamatot kívánunk összeállítani valamely cél elérése érdekében, az ismert építőelemekből. A folyamatok modelljeiben hagyományosan azok folyamatos működését feltételezik egyensúlyi állapotukban. A folyamatszintézisre kidolgozott eszköztár alkalmas szakaszos vagy egyszeri folyamatok vizsgálatára is, ekkor azonban az események időbeliségének figyelembe vételére is szükség van. Folyamatszintézis során használható modellezési technika kerül bemutatásra az időkorlátok kezelésére. A bemutatásra kerülő módszerek részben a feladat átfogalmazásával, részben a megoldó algoritmusok kiterjesztésével, garantálják az időkorlátok teljesülését az eredményül adott alternatív megoldások mindegyikére. A kidolgozott modelleket és megoldó algoritmusokat olyan nevezetes logisztikai feladat optimalizálásával szemléltetem, mint például a kapacitásos fuvarszervezés.

6.1. A MILP modell

Az időkorláttal kibővített PNS (Time Constrained PNS, TCPNS) feladatban, az összes végcélt időben el kell érni, miközben az erőforrások időbeli rendelkezésre állása korlátozott, illetve az összes tevékenység időtartama a méretének fix részt tartalmazó lineáris függvénye ([12]). A fix részt tartalmazó lineáris költségfüggvényű paraméteres PNS feladatot négy paraméterrel egészítettük ki az időkoráltok definiálására, amelyek a következőek. A tf_i az egész részű, tp_i pedig az arányos részű komponense annak a függvénynek, amelyik becslést ad minden egyes tevékenység időtartamára a méretük alapján. Az Ut_j az egyes végcélok határidejét jelöli, amely a következőképpen

definiálható:

$$Ut_j = \begin{cases} \geq 0, & \forall m_j \in \mathcal{P} \\ \max\limits_{m_j \notin \mathcal{P}} \{Ut_j\}, & \textit{máskülönben} \end{cases}. \tag{6.1.1}$$

Az Lt_j pedig az egyes erőforrások legkorábbi rendelkezésre állását jelöli, amely a következőképpen definiálható:

$$Lt_j = \begin{cases} \geq 0, & \forall m_j \in \mathcal{R} \\ 0, & \textit{máskülönben} \end{cases}. \tag{6.1.2}$$

A cél az, hogy meghatározzuk azt az $(\boldsymbol{m}^*, \boldsymbol{o}^*, \mathbf{x}^*, z^*)$ hálózatot, ami kielégíti a 3.3.1 - 3.3.6 és a 6.1.3 - 6.1.5 egyenleteket és a z^* minimális, ahogy az a 3.3.7 egyenletben látható.

Bármely m_j entitás kizárólag a legkorábbi Lt_j rendelkezésre állása és a Ut_j határideje közötti időszakban használható fel, azaz:

$$\forall m_j \in \mathcal{M} : Lt_j \leq t_{m_j} \leq Ut_j \tag{6.1.3}$$

Az $o_i \in \boldsymbol{o}$ tevékenység t_{o_i} kezdőidőpontja nem előzheti meg semelyik m_j előfeltételének rendelkezésre állási t_{m_j} idejét, azaz:

$$o_i = (\alpha, \beta) \in \boldsymbol{o}, \forall m_j \in \alpha_i : t_{o_i} \geq t_{m_j} \tag{6.1.4}$$

Egy o_i tevékenység bármely m_j következményének rendelkezésre állási t_{m_j} ideje nem előzheti meg a tevékenység t_{o_i} kezdő idejétől számított időtartamának $(tf_i + tp_i * x_i)$ befejeztét, azaz:

$$o_i = (\alpha, \beta) \in \boldsymbol{o}, \forall m_j \in \beta_i : t_{m_j} \geq t_{o_i} + tf_i + x_i * tp_i \tag{6.1.5}$$

6.2. Az időkorlátok relaxációja

Az TCPNS relaxált modelljében a 6.1.5 egyenletet a 6.2.1 és a 6.2.2 egyenletek szerint relaxáljuk, ahol tb a leghosszabb határidőt jelöli a végcélokhoz definiált határidők között.

$$tb = \max_{m_j \in \mathcal{P}} \{Ut_j\} \tag{6.2.1}$$

$$o_i = (\alpha, \beta) \in o, \forall m_j in \beta_i : t_{m_j} \geq t_{o_i} + x_i * tp_i + y_i * (tb + tf_i) - tb \tag{6.2.2}$$

Az y_i bináris változó jelöli az o_i tevékenység struktúrába történő bevonását, ekkor az $y_i = 1$ értéket vesz fel, illetve kizárását, amikor az $y_i = 0$ értékű lesz. A megoldási folyamatban egy tevékenység struktúrába történő bevonásáról vagy kizárásáról való döntést megelőzően becslést végzünk a költség alsó korlátjára és az alternatív szkenáriók időtartamára minden egyes lépésben.

$$y_i = 0 \Rightarrow t_{m_j} \geq t_{o_i} - tb \tag{6.2.3}$$

$$y_i = 1 \Rightarrow t_{m_j} \geq t_{o_i} + x_i * tp_i + tf_i \tag{6.2.4}$$

A becsléshez az y_i döntési változó relaxálható a 0-1 intervallumon, de ekkor is kapcsolatban van a tevékenység megengedett kapacitásával (ha egy tevékenység nem valósul meg, akkor csak nulla lehet a kapacitása), amely összefüggés a 6.2.5 egyenlettel adható meg.

$$x_i \geq u_i * y_i \tag{6.2.5}$$

A 6.2.2 egyenletben található egyenlőtlenség megegyezik a 6.2.3 és a 6.2.4 egyenletekkel attól függően, hogy az y_i egyenlő nullával vagy egyel. A 6.2.2 egyenlet korlátai csak akkor vannak hatással, ha az y_i értéke közel van az egyhez a relaxált modellben.

6.3. Szemléltető példa

A most következő alfejezetben a már korábban a 4.5 alfejezetben bemutatott jármű hozzárendelési feladat, azaz VAP kibővített változatán mutatom be az időkorlátok kezelésével kibővített PNS feladatot és annak megoldását. A feladat három teljesítendő szállítást, azaz végcélt tartalmaz (P1_Almádi, P2_Székesfehérvár, P3_Tihany), és ehhez három különböző jármű áll rendelkezésre (R1, R2, R3), ahogy az a feladat maximális struktúráján látható, amelyet a 6.2 ábra szemléltet [2]. Az R1-es teherautó Balatonkenesén állomásozik és méretéből és típusából adódóan Veszprémből Székesfehérvárra tud árut szállítani. Az R2-es teherautó Székesfehérváron állomásozik és árut szállíthat Veszprémből Tihanyra, illetve Tihanyból Balatonalmádiba. Az R3-as teherautó Tihanyban állomásozik és árut szállíthat Veszprémből Tihanyra, Veszprémből Székesfehérvárra, illetve Tihanyból Balatonalmádiba. A feladat kombinatorikus komponensei egy hármassal $(\mathcal{P}, \mathcal{R}, \mathcal{O})$ adhatóak meg, ahogy az a 2.2.3 fejezetben látható, ahol

$$
\begin{aligned}
\mathcal{R} &= \{P1_Almádi,\ P2_Székesfehérvár,\ P3_Tihany\} \\
\mathcal{P} &= \{R1,\ R2,\ R3,\ EUR, CO_2\} \\
\mathcal{O} &= \{r3_Balatonkenese\text{-}Vp,\ r3_Vp\text{-}Szf,\ r2_Szf\text{-}Vp,\ r2_Szf\text{-}Tihany, \\
&\quad r2_Vp\text{-}Tihany,\ r2_Tihany\text{-}Almádi,\ r2_Almádi\text{-}Vp,\ r1_Tihany\text{-}Almádi, \\
&\quad r1_Tihany\text{-}Vp,\ r1_Almádi\text{-}Vp_2,\ r1_Vp\text{-}Szf,\ r1_Vp\text{-}Tihany, \\
&\quad r1_Tihany\text{-}Vp_2,\ r1_Szf\text{-}Vp,\ r1_Szf\text{-}Tihany,\ r1_Tihany\text{-}Tihany2,\ Travel_R3, \\
&\quad Travel_R2,\ Travel_R1,\ r1_Tihany\text{-}Vp_3,\ r1_Almádi\text{-}Vp,\ r1_start_Tihany2\} = \\
&\quad \{(\{R3,\ Km_R3\}, \{R3_Vp\}), (\{R3_Vp,\ Km_R3\}, \{P2_Székesfehérvár\}), \\
&\quad (\{Km_R2,\ R2\}, \{R2_Vp\}), (\{Km_R2,\ R2\}, \{R2_Tihany\}), \\
&\quad (\{Km_R2,\ R2_Vp\}, \{P3_Tihany,\ R2_Tihany\}), \\
&\quad (\{Km_R2,\ R2_Tihany\}, \{P1_Almádi,\ R2_Almádi\}), \\
&\quad (\{Km_R2,\ R2_Almádi\}, \{R2_Vp\}), (\{Km_R1,\ R1_Tihany2\}, \{P1_Almádi,\ R1_Almádi \\
&\quad (\{Km_R1,\ R1\}, \{R1_Vp_2\}), (Km_R1,\ R1_Almádi\}, \{R1_Vp_1), \\
&\quad (\{Km_R1,\ R1_Vp_1\}, \{P2_Székesférvár,\ R1_Szf\}), \\
&\quad (\{Km_R1,\ R1_Vp_2\}, \{P3_Tihany,\ R1_Tihany\}),
\end{aligned}
$$

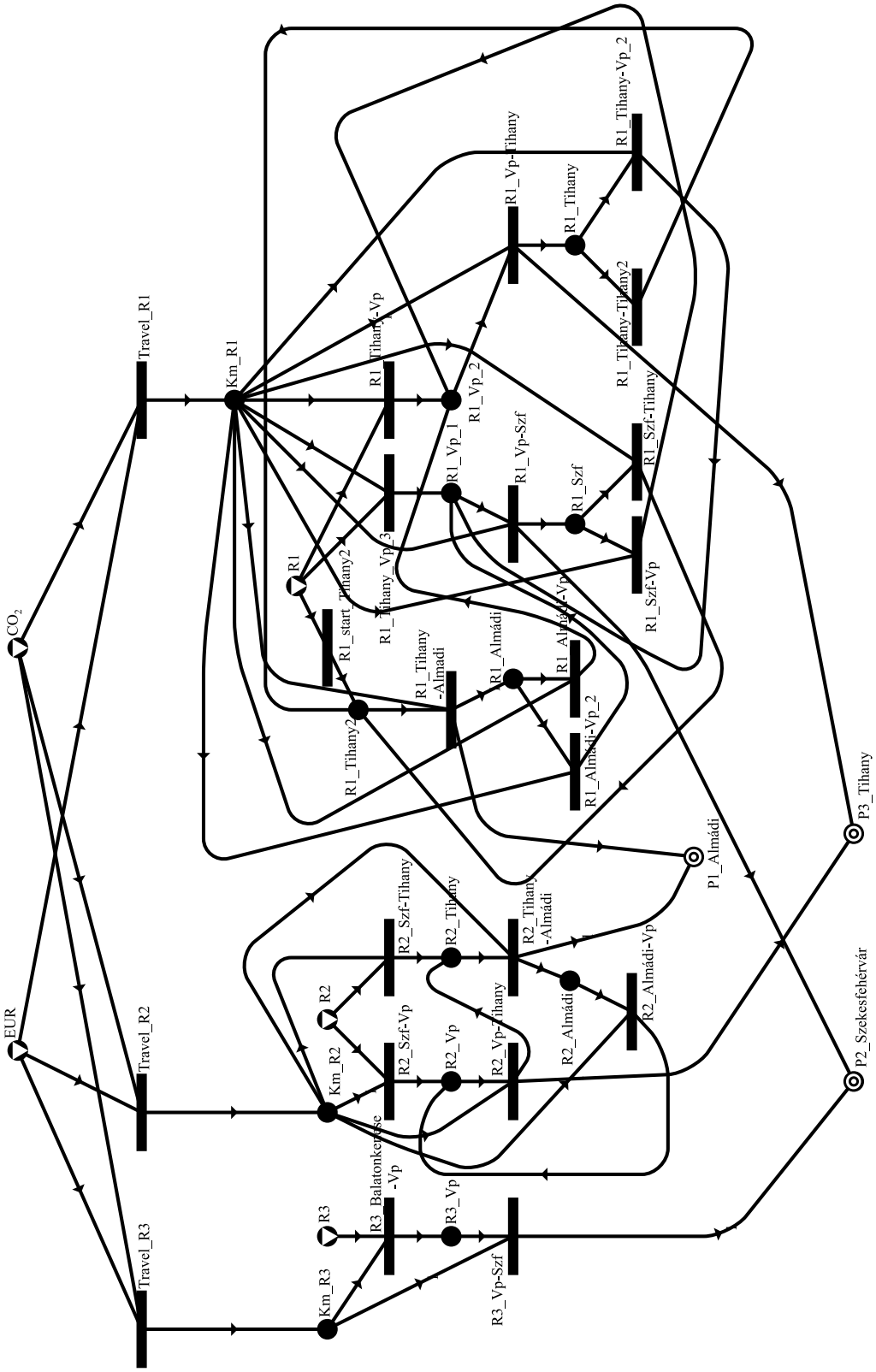
6.1. táblázat. Az entitások paraméterei

Entitások	**Típus**	**Legkorábbi rendelkezésre állás** (Lt) **[min]**	**Határidő** (tb) **[min]**
R1	erőforrás	20	0
R2	erőforrás	0	0
R3	erőforrás	0	0
EUR	erőforrás	0	0
CO_2	erőforrás	0	0
P1_Almádi	végcél	0	180
P2_Székesfehérvár	végcél	0	160
P3_Tihany	végcél	0	130

$$(\{Km_R1,\ R1_Tihany\}, \{R1_Vp_1\}), (\{Km_R1,\ R1_Szf\}, \{R1_Vp_2\}),$$
$$(\{Km_R1,\ R1_Szf\}, \{R1_Tihany2\}), (\{Km_R1,\ R1_Tihany\}, \{R1_Tihany2\}),$$
$$(\{EUR,\ CO_2\}, \{Km_R3\}), (\{EUR,\ CO_2\}, \{Km_R2\})$$
$$(\{EUR,\ CO_2\}, \{Km_R1\}), (\{Km_R1,\ R1\}, \{R1_Vp_1\})$$
$$(\{Km_R1,\ R1_Almádi\}, \{R1_Vp_2\}), (\{Km_R1,\ R1\}, \{R1_Tihany2\})\}$$

A feladat entitásainak idő paramétereit a 6.1. táblázat, a tevékenységeinek idő paramétereit pedig a 6.2. táblázat foglalja magába. Ahogy az a 6.1. táblázatban látható, három végcélt tartalmaz a feladat a *P1_Almádi*-t, amit 180 perc alatt, a *P1_Székesfehérvár*-t, amit 160 perc alatt és a *P1_Tihany*-t, amit 130 perc alatt kell teljesíteni.

Felhasználva a 6.2 fejezetben bevezetett időkorlát relaxációt a feladat megoldható, amelynek optimális megoldás struktúrája a 6.2 ábrán látható. Azaz az R1-es és az R3-as teherautókkal teljesíthető mind a három szállítási feladat a megadott határidőkön belül. Az R1-es autó Balatonkeneséről indulva 160 perc alatt ér Székesfehérvárra (P1_Székesfehérvár) a Veszprémből szállított áruval, azaz pont teljesíti a megadott határidőt. Az R2-es autó Tihanyból kiindulva először Veszprémből Tihanyba (P3_Tihany) szállít árut az indulástól számított 130 percen belül, majd pedig



6.1. ábra. Az időkorlátokkal bővített jármű hozzárendelési feladat maximális struktúrája

6.2. táblázat. A tevékenységek előfeltételei, következményei és idő paraméterei

Tevékenységek	**Fix idő** (tf) **[min]**	**Arányos idő** (tp) **[min]**
r3_Balatonkenese-Vp	60	0
r3_Vp-Szf	80	0
r2_Szf-Vp	80	0
r2_Szf-Tihany	140	0
r2_Vp-Tihany	60	0
r2_Tihany-Almádi	40	0
r2_Almádi-Vp	50	0
r1_Tihany-Almádi	40	0
r1_Tihany-Vp	60	0
r1_Almádi-Vp_2	50	0
r1_Vp-Szf	80	0
r1_Vp-Tihany	60	0
r1_Tihany-Vp_2	60	0
r1_Szf-Vp	80	0
r1_Szf-Tihany	140	0
r1_Tihany-Tihany2	0	0
Travel_R3	0	0
Travel_R2	0	0
Travel_R1	0	0
r1_Tihany-Vp_3	60	0
r1_Almádi-Vp	50	0
r1_start_Tihany2	0	0

Tihanyból Balatonalmádiba (P2_Balatonalmádi) az indulástól számított 160 percen belül, azaz mindegyik szállítás teljesíti a kitűzött határidőket.

6.4. Strukturális kiegészítés az előidejűség kezelésére

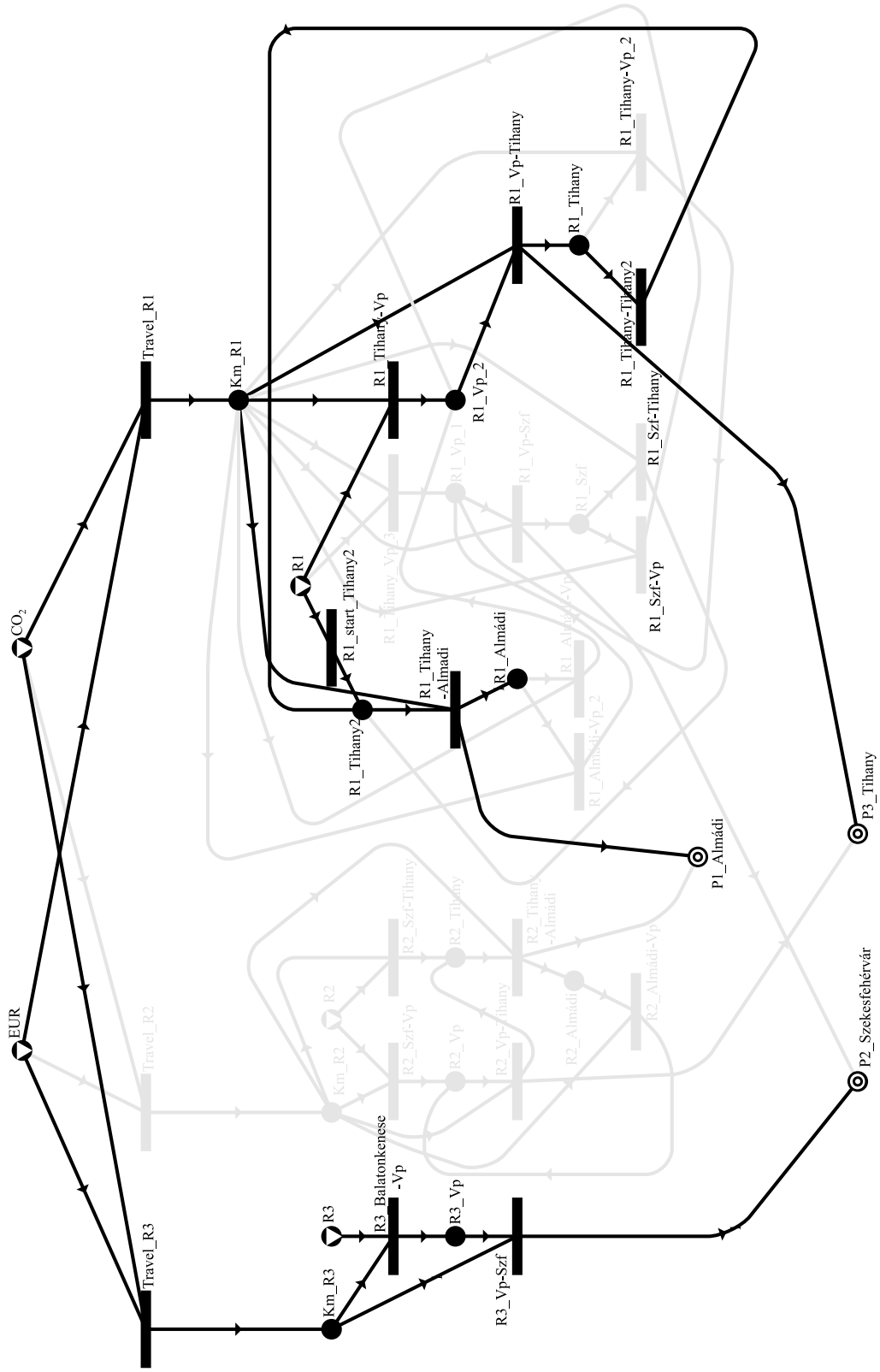
Az előidejűség azaz a precedencia feladata egy meghatározott sorrend biztosítása. Egy P_i folyamat T_i és egy P_j folyamat T_j tevékenységeinél a precedencia akkor áll fenn, ha T_j végrehajtása csak akkor kezdődhet el, ha a T_i tevékenység már befejeződött. A folyamat szintézisbe történő időkorlátok bevezetésével, a strukturális modellt ki kell bővíteni azért, hogy kifejezze vajon egy tevékenység következménye és egy másik tevékenység előfeltétele megelőzi-e egymást vagy sem. A maximális struktúrába ehhez mesterséges tevékenységek bevezetése szükséges, mint például a szemléltető példában az $ot1$ és $ot2$ tevékenységek a 6.3. ábrán. Meg kell azonban jegyezni, hogy ezen mesterséges tevékenységek bevonása vagy kizárása a megoldás struktúrából egyértelműen szemlélteti a tevékenységek relatív sorrendjét, ahogy az a 6.3. ábrán is látható. Ezen kiegészítő tevékenységek a hálózat működésre semmilyen hatással nincsenek, csak a előidejűség miatti szeparálódást segítik elő.

A 6.3. ábrán látható szemléltető példán a tb határidő, akkor és csak akkor teljesíthető, ha az $m5$ és az $m6$ végcélok függetlenek. Ez olyan megoldás struktúrával érhető el, amelyikben az $m4$, az $o4$ tevékenység előfeltétele, kizárólag az $o2$ tevékenység által biztosított, azaz az $o4$ nem várja meg míg az $o1$ előállítja a kimenetét.

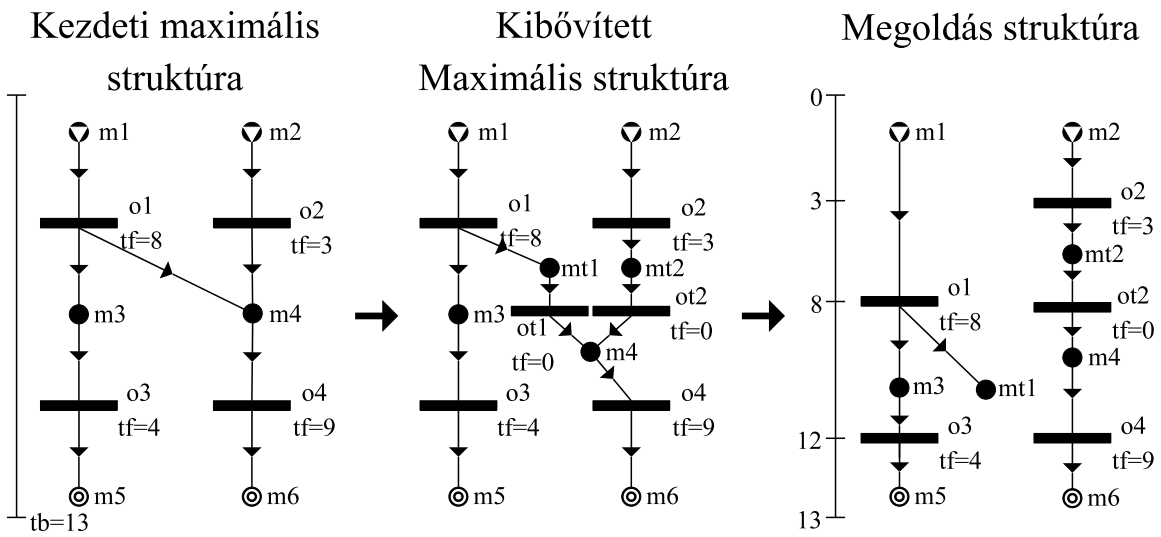
A 6.4. ábrán látható MSETC (Maximal Structure Extender for Time Constarints) algoritmus szolgál a maximális struktúra kibővítésére az előidejűséget szemléltető kiegészítő tevékenységekkel. Az algoritmus sorra megvizsgálja az egyes entitásokat, és amelyiket egynél több tevékenység eredményezheti, annál az őt eredményező tevékenységek és az entitás közé betesz egy-egy kiegészítő tevékenységet a hozzá tartozó kiegészítő entitással.

Tétel 6.4.1 *Az MSETC algoritmus futási ideje polinomiális.*

Bizonyítás *A bizonyítás során a „legrosszabb eset" kifejezést mindig a futási idő szempontjából tekintem.*



6.2. ábra. Az időkorlátokkal kibővített jármű hozzárendelési feladat megoldás struktúrája



6.3. ábra. Maximális struktúra előidejűség szerinti kiterjesztése az időkorlátok kezelésére

```
01.          function MSETC((m, o): P-graph): extended maximal structure
02.          begin
03. 1. lépés   for all m_j ∈ m begin
04. 2. lépés       if |φ⁻({m_j})| > 1 then
05. 4. lépés           for all o_i1 ∈ φ⁻({m_j}) begin
06. 5. lépés               m := m ∪ {m_jti1};
07. 6. lépés               o_i1 := (ψ⁻(o_i1), ψ⁺(o_i1) ∪ {m_jti1} \ {m_j});
08. 7. lépés               for all o_i2 ∈ φ⁺({m_j}) begin
09. 8. lépés                   m := m ∪ {m_jti2};
10. 9. lépés                   o_i2 := (ψ⁻(o_i2) ∪ {m_jti2} \ {m_j}, ψ⁺(o_i2));
11. 10. lépés                  o := o ∪ {({m_jti1}, {m_jti2})};
12.                        end;
13.                    end;
14.                    m := m \ {m_j};
15.                end if;
16.            end;
17.            return (m, o) ;
18.          end;
```

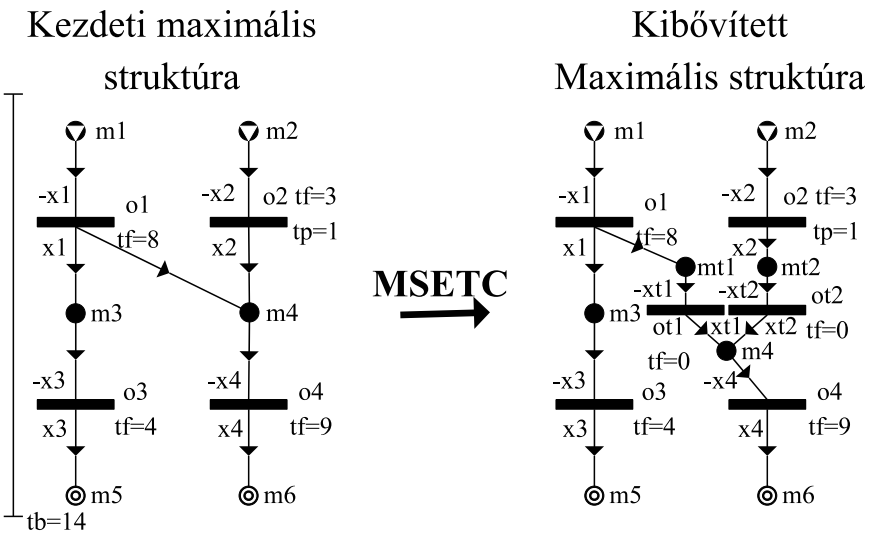
6.4. ábra. Az MSETC algoritmus

Az MSETC algoritmus három egymásba ágyazott ciklust tartalmaz, amely meghatározza a futási idejét. Az algoritmus keretét adó ciklus – amelyet az 1. lépés azonosít – sorra veszi az összes m_j entitást. Ha a 2. lépésben található feltétel mindegyik entitásra teljesül, akkor a 4. lépésben sorra veszi az összes tevékenységet, aminek m_j eredménye, majd a 7. lépésben minden tevékenységet, aminek m_j előfeltétele. Nyilvánvaló módon, egyik sem lehet több , mint az összes akció száma, azaz a ciklus legfeljebb $|\mathcal{m}|^2$ iterációt végezhet. Könnyen belátható, hogy a halmazok és relációk megfelelő ábrázolásával az 5,6,8,9,10 lépések mindegyike külön-külön és együtt is megvalósíthatóak $O(|\mathcal{m}| + |\mathcal{o}|)$ lépésben. Az előzőekben bemutatottakat összegezve tehát az MSETC algoritmus $O(|\mathcal{m}| * |\mathcal{o}|^2 * (|\mathcal{m}| + |\mathcal{o}|))$ futási idejű.

6.5. Az előidejűség kezelését szemléltető példa

Az előző fejezetben bemutatott példa részletesebb, módosított kifejtésével – amely a 6.5. ábrán látható – fogjuk az újonnan bevezetett P-gráf módszertanbeli előidejőség kezelést szemléltetni. A feladat kombinatorikus komponensei egy hármassal $(\mathcal{P}, \mathcal{R}, \mathcal{O})$ adhatóak meg, ahogy az a 2.2.3. fejezetben látható, ahol
$\mathcal{R} = \{m1, m2\}$
$\mathcal{P} = \{m5, m6\}$
$\mathcal{O} = \{o1, o2, o3, o4\} = \{(\{m1\}, \{m3,\ m4\}),\ (\{m2\},\ \{m4\}),\ (\{m3\}, \{m5\}),\ (\{m4\},\ \{m6\})\}$

Ahogy az a 6.5. ábrán látható, a feladat maximális struktúráját ki kell bővíteni az MSETC algoritmus segítségével az előidejűség kezelését lehetővé tevő tevékenység(ek)el. Jelen feladat az $ot1$ és $ot2$ tevékenységekkel, és a hozzájuk tartozó $mt1$ és $mt2$ entitásokkal lett kibővítve és a feladat kombinatorikus komponensei $(\mathcal{P}, \mathcal{R}, \mathcal{O})$ a következőképpen változnak.
$\mathcal{R} = \{m1, m2\}$
$\mathcal{P} = \{m5, m6\}$
$\mathcal{O} = \{o1, o2, o3, o4, ot1, ot2\} = \{(\{m1\}, \{m3, mt1\}),\ (\{m2\}, \{mt2\}),\ (\{m3\},\ \{m5\}),\ (\{m4\},\ \{m6\}),\ (\{mt1\},\ \{m4\}),\ (\{mt2\},\ \{m4\})\}$



6.5. ábra. A szemléltető példa maximális struktúrájának előidejűség szerinti kiterjesztése az időkorlátok kezelésére

A feladat entitásainak paramétereit a 6.3. táblázat, a tevékenységeinek paramétereit pedig a 6.4. táblázat foglalja magába. Ahogy az a 6.3 táblázatban látható, kettő végcélt tartalmaz a feladat, az $m5$-öt, amit 13 óra alatt és az $m6$-ot, amelyet 14 óra alatt kell teljesíteni.

A 6.6. ábrán látható a feladat megoldásához vezető döntési fa. A feladat megoldása során 8 állapotot és az ezekhez tartozó 7 döntést kell megvizsgálni, amíg eljutunk a feladat megoldásstruktúrájához. Az egyes állapotokat a sorszámukkal ellátott teli

6.3. táblázat. Az entitások paraméterei

Entitások	**Típus**	**Legkorábbi rendelkezésre állás** (Lt) **[h]**	**Határidő** (tb) **[h]**
m1	erőforrás	0	0
m2	erőforrás	1	0
m5	végcél	0	13
m6	végcél	0	14

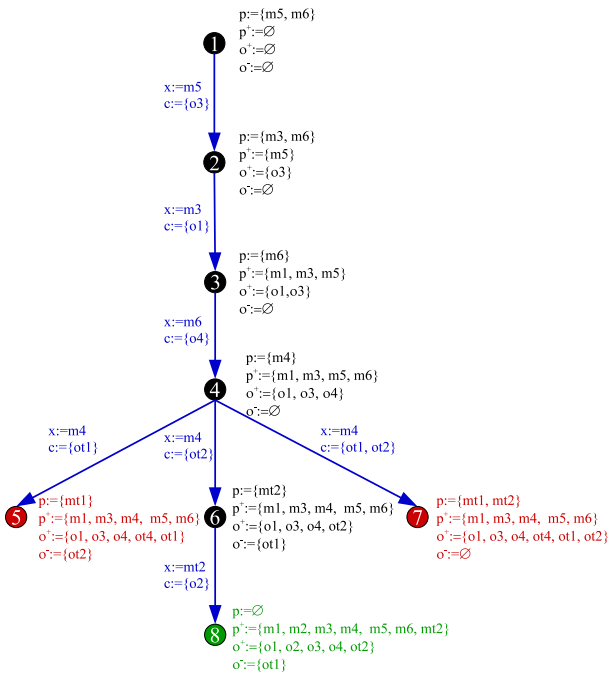
6.4. táblázat. A tevékenységek előfeltételei, következményei és idő paraméterei

Tevékenységek	**Előfeltételek**	**Következmények**	**Fix idő** (tf) **[h]**	**Arányos idő** (tp) **[h]**
o1	m1	m3, m4	8	0
o2	m2	m4	3	1
o3	m3	m5	4	0
o4	m4	m6	9	0

körrel jelöltük. A fekete színűek az egyes köztes állapotokat jelölik, a pirosak a valamilyen szempont szerint megoldhatatlan struktúrákat, a zöld pedig a minden feltételnek megfelelő megoldásstruktúrát.

A p halmaz azokat az entitásokat tartalmazza, amelyekről még nem született döntés, hogy bekerülnek-e a megoldás struktúrába, ezzel együtt a p^+ halmazba vagy sem. Ha nem kerül be akkor átkerül a p^- kizárt entitások halmazába. A kiválasztott entitás vizsgálatával egy időben döntünk arról, hogy az őt előállító tevékenységek szintén bekerülnek-e az adott részstruktúrába, azaz az o^+ halmazba vagy kizárjuk őket és ebben az esetben pedig az o^- halmazba kerülnek.

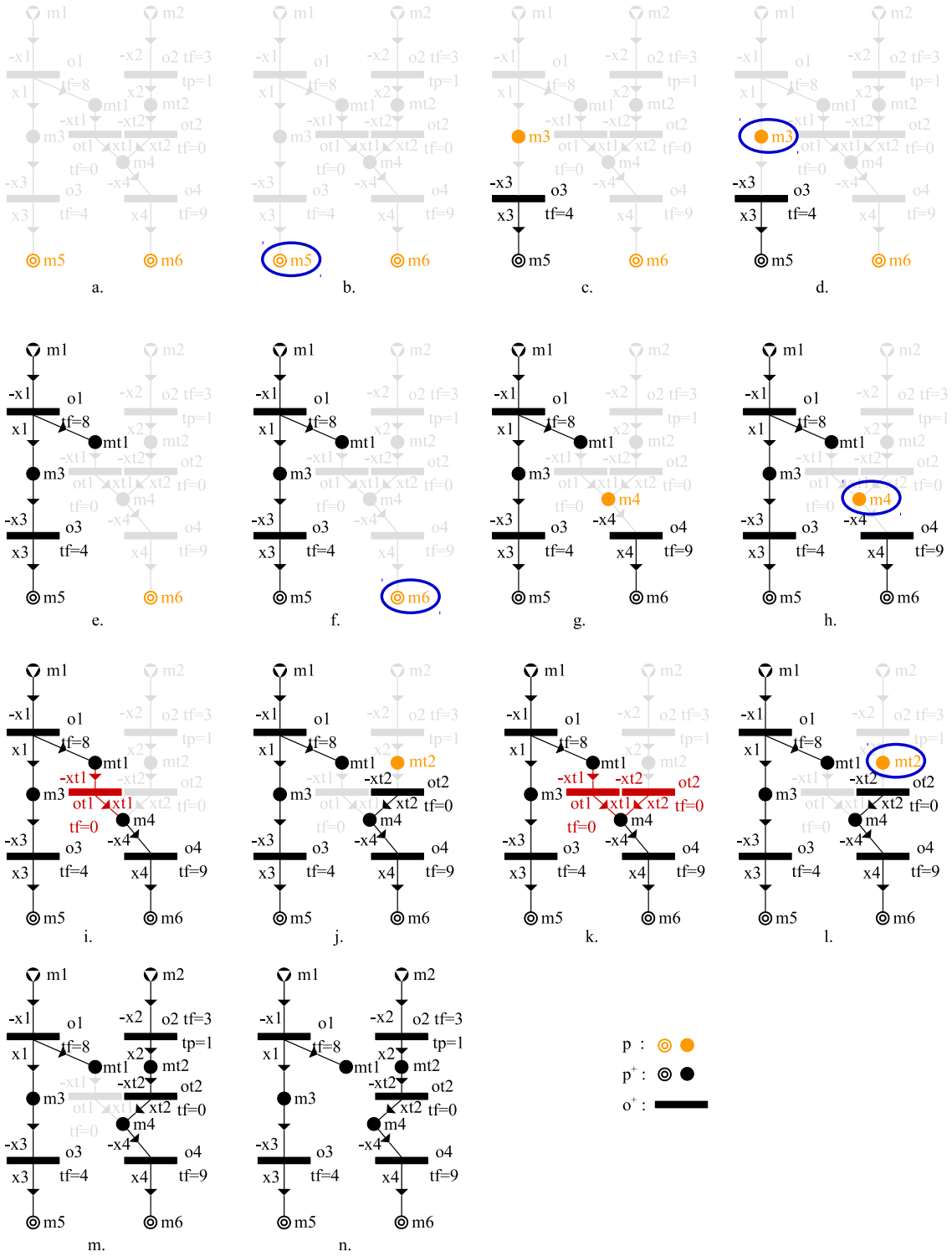
A 6.7. ábra az előbbiekben említett állapotokat és döntéseket szemlélteti lépésről lépésre. A megoldás a 6.7.a. ábrán látható előidejűség kezelésével kibővített maximális struktúrából kiindulva a végcélok meghatározásával kezdődik. Ezek az $m5$ és az $m6$, amelyek mindegyike belekerül a p halmazba, ahogy azt a döntési fa 1. állapota is mutatja. A 6.7.b. ábrán láthatóan a p halmazból kiválasztjuk az $m5$ entitást, hogy megvizsgáljuk bekerülhet-e a megoldásstruktúrába az őt előállító $o3$ tevékenységgel együtt. Mivel az adott részstruktúra minden feltételt maradéktalanul teljesít, így bekerül az $m5$ a p^+ halmazba, illetve az $o3$ az o^+ halmazba, továbbá az $o3$ előfeltétele az $m3$ bekerül a p halmazba, ahogy az a 6.7.c. ábrán és a döntési fa 2. állapotában látható. Ezek után a p halmazból kiválasztjuk az $m3$ entitást, hogy ebben az esetben is megvizsgáljuk bekerülhet-e a megoldásstruktúrába az őt előállító $o1$ tevékenységgel együtt, továbbá mivel az $o1$ előfeltétele az $m1$ entitás egy erőforrás, ezért annak a struktúrába való bevonásáról is döntést hozunk, ahogy az a 6.7.d. ábrán látható.



6.6. ábra. A feladat megoldásához vezető döntési fa

Mivel az adott részstruktúra minden feltételt maradéktalanul teljesít így bekerül az $m3$ és az $m1$ a p^+ halmazba, illetve az $o1$ az o^+ halmazba, ahogy az a 6.7.e. ábrán és a döntési fa 3. állapotában látható. Majd ezután a p halmazból kiválasztjuk az $m6$ entitást, hogy megvizsgáljuk bekerülhet-e a megoldásstruktúrába az őt előállító $o4$ tevékenységgel együtt, ahogy az a 6.7.f. ábrán látható. Mivel az így kapott részstruktúra minden feltételt maradéktalanul teljesít így bekerül az $m6$ a p^+ halmazba, illetve az $o4$ az o^+ halmazba, továbbá az $o4$ tevékenység előfeltétele az $m4$ bekerül a p halmazba, ahogy az a 6.7.g. ábrán és a döntési fa 4. állapotában látható.

Majd ezután a p halmazból kiválasztjuk az $m4$ entitást, hogy megvizsgáljuk bekerülhet-e a megoldásstruktúrába, ahogy ezt a 6.7.h. ábra szemlélteti. Mivel az $m4$ entitást két tevékenység, az MSETC algoritmus által bevezetett $ot1$ és $ot2$ is eredményezni tudja, ezért ebből az állapotból 3 ($2^2 - 1 = 3$) úton is tovább haladhat a megoldó algoritmus, ahogy azt a 6.6. ábrán látható döntési fa is szemlélteti. Az első lehetőség a döntési fa 5. állapota, amely esetben csak az $ot1$ tevékenységet próbáljuk a megoldásstruktúrába bevonni, ahogy azt a 6.7.i. ábra mutatja. Mivel ebben az



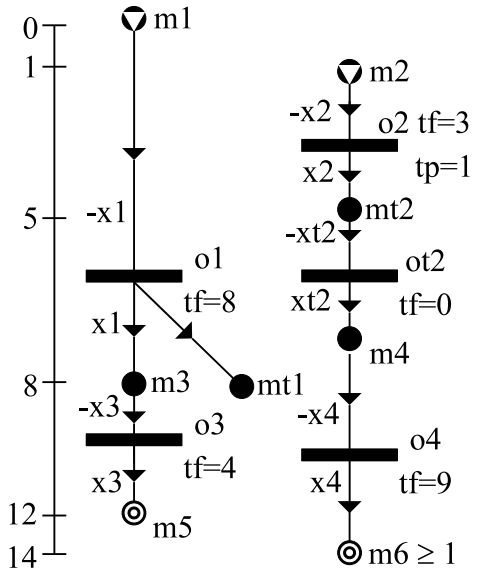
6.7. ábra. A megoldásstruktúrához vezető döntések szemléltetése

$t_{m1} \geq 0 < 14$ ✔
$t_{o1} \geq t_{m1} = 0 < 14$ ✔
$t_{m3} = t_{mt1} \geq t_{o1} + tf_{o1} + tp_{o1} * x1 = 0 + 8 + 0 = 8 < 14$ ✔
$t_{o3} \geq t_{m3} = 8 < 14$ ✔
$t_{m5} \geq t_{o3} + tf_{o3} + tp_{o3} * x3 = 8 + 4 + 0 = 12 < 14$ ✔
$t_{ot1} \geq t_{mt1} = 8 < 14$ ✔
$t_{m4} \geq t_{ot1} + tf_{ot1} + tp_{ot1} * x_{ot1} = 8 + 0 + 0 = 8 < 14$ ✔
$t_{o4} \geq t_{m4} = 8 < 14$ ✔
$t_{m6} \geq t_{o4} + tf_{o4} + tp_{o4} * x4 = 8 + 9 + 0 = 17 > 14$ ✘

6.8. ábra. A döntési fa 5. állapotában az időkorlát túllépésének szemléltetése

esetben az $m6$ végcél csak 17 óra alatt állhat elő, ami nem teljesíti a 14 órás időkorlátot, ahogy az a 6.8. ábrán látható, ezért a döntési fa ezen ága nem képes minden feltételt maradéktalanul kielégítő megoldást szolgáltatni, ezért további vizsgálata nem szükséges és az adott részstruktúrát elhagyhatjuk. Az második lehetőség a döntési fa 6. állapota, amely esetben csak az $ot2$ tevékenységet próbáljuk a megoldásstruktúrába bevonni, ahogy azt a 6.7.j. ábra mutatja. Az így kapott részstruktúra minden feltételt maradéktalanul teljesít így bekerül az $m4$ a p^+ halmazba, illetve az $ot2$ az o^+ halmazba, továbbá az $o4$ tevékenység előfeltétele az $mt2$ bekerül a p halmazba. Az harmadik lehetőséget a döntési fa 7. állapota szemlélteti, amely esetben az $ot1$ és az $ot2$ tevékenységeket együtt próbáljuk a megoldásstruktúrába bevonni, ahogy azt a 6.7.k. ábra mutatja. Mivel ebben az esetben az $m6$ végcél szintén csak 17 óra alatt állhat elő, a 6.2.1 és a 6.1.1 egyenletek következtében. Így nem teljesül a 14 órás időkorlát, ezért a döntési fa ezen ága nem képes minden feltételt maradéktalanul kielégítő megoldást szolgáltatni, ezért további vizsgálata nem szükséges, és az adott részstruktúrát elhagyhatjuk.

Végezetül az 6.7.h. ábra szerint a p halmazból kiválasztjuk az $mt2$ entitást, hogy megvizsgáljuk bekerülhet-e a megoldásstruktúrába az őt előállító $o2$ tevékenységgel illetve az $m2$ – mivel az $o2$ tevékenység $m2$ előfeltétele erőforrás – entitással együtt. Mivel az így kapott részstruktúra minden feltételt maradéktalanul teljesít, így bekerül az $mt2$ és az m2 a p^+ halmazba, illetve az $o2$ az o^+ halmazba, ahogy az a 6.7.m. ábrán és a döntési fa 8. állapotában látható. Ahogy azt a 6.7.n. ábra jól szemlélteti, a



6.9. ábra. Az előidejűséggel bővített időkorlát kezelést szemléltető példa megoldás struktúrája

hálózat szétbomlik két külön álló részre, hogy teljesülhessenek a kitűzött időkorlátok.

A feladat bonyolultságát az adja, hogy minden entitás bevárja az összes őt eredményező tevékenység teljesülését. Az $m5$ végcél egyértelműen csak az $o1$ és az $o2$ tevékenységek által állhat elő, míg az $m6$-hoz az $o4$ közbeiktatásával az $o1$ és az $o2$ tevékenységen keresztül is vezet út. Mivel az $o2$-es tevékenységnek mindenképpen szerepelnie kell a megoldásban, hogy az $m6$ teljesíthető legyen és az $m4$ mind az $o1$ és mind az $o2$ tevékenységek befejeztét bevárja, mielőtt felhasználható lenne az $o4$-es tevékenységhez, ezért csak a nyolcadik óra után használható fel az $o4$-es által. Így viszont a $m6$-os végcél soha nem állhatna elő a 14 órás határidőn belül, mivel az $o4$-es tevékenység 9 órán át tart $(8 + 9 > 14)$. Viszont, ha csak a $o2$-es tevékenységet vennénk figyelembe az $m4$-es entitás elérésénél, akkor teljesíthető lenne az $m6$-os tevékenység határideje $(1 + (3 + 1) + 9 = 14)$. Ezért a MSETC algoritmust felhasználva az $m4$-es entitás és az őt eredményező $o1$ és $o2$ tevékenységek közé bekerülnek az $mt1$ és az $mt2$ kiegészítő entitások és az $ot1$ és az $ot2$ kiegészítő tevékenységek ahogy az az 6.5 ábrán látható. Így már szét tud válni úgy a hálózat, hogy az $o1$ befejezte előtt aktiválódjon az $o4$, ahogy az az 6.9 ábrán látható. Tehát az MSETC algoritmus

segítségével megoldhatóvá vált ez az előidejűség problémáját felvető feladat.

6.6. A TCPNS-ben rejlő továbblépési lehetőség

A TCPNS-ben azon túl, hogy többcélú – idő és költség szerinti – optimalizálásra használható, számos további lehetőség rejlik. Az egyik ilyen lehetőség az ütemezés. Jelen dolgozatnak nem tárgya ezen továbblépési lehetőség részletes bemutatása, csupán egy kitekintést szeretnénk adni róla, ezzel is hangsúlyozva a jelen kutatási eredmények fontosságát.

Termelő és szolgáltató rendszerek működését, viselkedését elemezve az esetek jelentős részében ütemezési feladatokkal találkozunk, így például a vegyiparban, az olajiparban, a gépiparban, a mezőgazdaságban, az építőiparban és a szállítmányozásban. A termelő és szolgáltató rendszerek mellett az informatikai rendszerekhez is kapcsolódnak bonyolult ütemezési feladatok. Ha a rendszer tartalmaz olyan konkurens folyamatokat, melyek ugyanazokat az erőforrásokat igénylik és nem áll rendelkezésre megfelelő számú erőforrás, akkor az erőforrások ütemezésével lehet a folyamatokat kiszolgálni.

Az ütemezési feladatot a megfelelő folyamathálózat szintézis feladattá átfogalmazva, az a P-gráf módszertan algoritmusaival különböző szempontok szerint megoldhatóvá válik, amely számtalan új lehetőséget nyithat meg előttünk. Az átalakítás után a berendezések lesznek az erőforrások és a cél pedig annak a struktúrának a meghatározása, amely az előírt végcélokat képes elérni a receptgráf alapján. Az ütemezési feladatok két leggyakoribb célja a teljes működési idő minimalizálása, illetve a profit maximalizálása. Észrevehetjük, hogy e két cél szerinti optimalizálást egyszerre elégíti ki a TCPNS.

Számos módszer létezik ütemezési feladtok megoldásra, ilyenek például az S-gráf vagy a Slot MILP modellt (például a korábban tárgyalt STN, RTN módszertanok), illetve a Precedencia alapú MILP modellt használó módszertanok. A 6.5. táblázat foglalja össze ezeknek a módszertanoknak az egymással és a TCPNS-el történő összehasonlítását. Ahogy az látható a 7 összehasonlítási szempontból a TCPNS mindegyiknek eleget tesz, amíg a többi módszertan mindegyike nagyjából a felének, amely

6.5. táblázat. A különböző módszerek összehasonlítása

	Slot MILP modell	**Precedencia alapú MILP modell**	**S-gráf**	**TCPNS**
Automatikus modell generálás	✔		✔	✔
MILP modell	✔	✔		✔
Gráf reprezentáció	✔		✔	✔
Kombinatorikus gyorsítás			✔	✔
Elosztott betöltés		✔		✔
Vegyes tárolási stratégia	✔	✔		✔
Többcélú optimalizálás				✔

jól szemlélteti, hogy milyen gazdag eszköztárral rendelkezik az újonnan kidolgozott TCPNS.

Mint ahogy azt a fejezet elején említettem, ez a továbblépési lehetőség nem tárgya a jelen dolgozatnak, ezért részletesen nem is tárgyaljuk. Csupán arra szerettük volna felhívni a figyelmet, hogy az újonnan kidolgozott módszer további kutatási témák alapjául szolgálhat.

6.7. Összefoglalás

Kibővítettem a PNS feladat megfogalmazását az erőforrások időbeli rendelkezésre állásának kezelésével, a végcélok keletkezési határidejét tartalmazó paraméterekkel és a tevékenységek végrehajtásához szükséges időt becslő függvények paramétereivel.

Olyan bizonyítottan polinomiális futási idejű algoritmust dolgoztam ki, amely a folyamat-hálózat maximális struktúrájában megjelenő alternatív tevékenységek esetén azok időbeli szinkronizálását kifejező kiegészítő tevékenységekkel bővíti a maximális struktúrát. Így lehetőséget adva a különböző megengedett struktúrákban az alternatív tevékenységek szinkron vagy aszinkron beépítésere.

Megmutattam, hogy a maximális struktúra kibővítése után a folyamat-szintézis feladat megfelelő MILP formalizálásával a szinkron és az aszinkron működésű megoldások egyaránt megtalálhatóak matematikai programozással is.

A fejezetben bemutatott – általam kidolgozott – módszereket szemléltető példákon is demonstráltam, továbbá implementáltam őket IBM ILOG CPLEX Optimization Studioban és Zimpl-ben, amely implantációkkal számos egyéb feladatot is megoldottam.

6.8. Kapcsolódó publikációim

Referált nemzetközi folyóirat cikkek

1. Kalauz, K., Süle, Z., Bertók, B., Friedler, F. és Fan, L. T. Extending process-network synthesis algorithms with time bounds for supply network design, *Chemical Engineering Transactions*, 29:259-264, 2012.

Nemzetközi konferencia előadások

2. Kalauz, K., Süle, Z., Bertók, B., Friedler, F., and Fan L. T., Extending process-network synthesis algorithms with time bounds for supply network design, PRES 2012, Praha, Czech Republic, August 25-29, 2012.

1. Bertók, B., Kalauz, K., Süle, Z., Friedler, F., and Fan L. T., Algorithmic Synthesis of Process Networks wit Time Constraints by the P-garph Framework, VOCAL 2010, Veszprém, Hungary, December 13-15, 2010.

7. fejezet

Összefoglalás

A dolgozat 4. fejezetében bemutattam, hogy olyan folyamathálózat-szintézis feladatok esetén, ahol a tevékenységek be- és kimenetei – összekapcsolhatóságukat meghatározó tulajdonságuk alapján – véges sokfélék lehetnek, a Friedler és szerzőtársainak munkáiból ismert, folyamathálózatok szintézisére kidolgozott eszközök jól adaptálhatóak. A szintézis feladat véges halmazokkal meghatározható. A megengedett vagy lehetséges megoldások struktúrája P-gráffal leírható. Ezen leírás alapján strukturális tulajdonságaik egyértelműen meghatározhatóak. A keresési térnek a strukturálisan megengedett hálózatok körére való leszűkítése a feladat algoritmikus megoldását nagyban elősegíti. Továbbá bemutattam milyen megfontolások miatt szükséges a PNS keretrendszer kiterjesztése a kölcsönösen kizáró halmazok kezelésére, és hogy milyen változtatásokat kellett elvégezni az eredeti P-gráf keretrendszeren, hogy annak algoritmusai hatékonyan figyelembe tudják venni ezeket az új megkötéseket. Bevezettem egy új strukturális feltételt, amely biztosítja, hogy egyetlen egy megoldás struktúra sem tartalmaz kölcsönösen kizáró tevékenységeket. Az új korlátozások bevezetése a keresési tér további csökkenését eredményezheti, amelyet az algoritmusok – a dolgozatban bemutatott – módosításával hatékonyan ki lehet használni a megoldási idő csökkentésére, valamint megoldás során generált részproblémák számának csökkentésére.

Az 5. fejezetben ismertettem a redundanciában rejlő potenciális lehetőségeket és

előnyöket. Bevezettem a megbízhatóság kiszámítási módszerét a P-gráf módszertanban. Algoritmikus módszert dolgoztam ki a szuperstruktúra olyan kiterjesztésére, amely garantálja a redundanciát és azáltal növeli a teljes megbízhatóságát egy ellátási szcenáriónak. A redundánsan kiterjesztett maximális struktúrát alapul véve az összes szcenárió generálható az eredeti P-gráf algoritmusokkal.

Végül, a 6. fejezetben a PNS feladat megfogalmazását kibővítettem az erőforrások időbeli rendelkezésre állásának kezelésével, a végcélok keletkezési határidejét tartalmazó paraméterekkel és a tevékenységek végrehajtásához szükséges időt becslő függvények paramétereivel. Olyan bizonyítottan polinomiális futási idejű algoritmust dolgoztam ki, amely a folyamat-hálózat maximális struktúrájában megjelenő alternatív tevékenységek esetén azok időbeli szinkronizálását kifejező kiegészítő tevékenységekkel bővíti a maximális struktúrát. Így lehetőséget adva a különböző megengedett struktúrákban az alternatív tevékenységek szinkron vagy aszinkron beépítésere. Megmutattam továbbá, hogy a maximális struktúra kibővítése után a folyamat-szintézis feladat megfelelő MILP formalizálásával a szinkron és az aszinkron működésű megoldások egyaránt megtalálhatóak matematikai programozással is.

8. fejezet

Saját publikációk

Referált nemzetközi folyóirat cikkek

5. Bertók, B., Kalauz, K., Sule, Z., and Friedler, F., Combinatorial Algorithm for Synthesizing Redundant Structures to Increase Reliability of Supply Chains: Application to Biodisel Suplly, *Industrial & Engineering Chemistry Research*, 52(1), 181–186, 2013.

4. Kalauz, K., Süle, Z., Bertók, B., Friedler, F., and Fan, L. T., Extending process-network synthesis algorithms with time bounds for supply network design, *Chemical Engineering Transactions*, 29, 259–264, 2012.

3. Szlama, A. G., Kalauz, K., Bertók, B., and Heckl, I., Solving a separation-network synthesis problem adopting interval optimization techniques, *Chemical Engineering Transactions*, 29, 1525-1530, 2012.

2. Heckl, I., Kalauz, K., Kalocsai, P. and Halász, L., Custom simulator for logistic networks in downstream, *Clean Technology and Environmental Policy*, 12, 627-634, 2010.

1. Heckl, I., Kalocsai, P. and Halász, L., Kalauz, K., Event driven process simulation of pipeline networks, *Chemical Engineering Transactions*, 18, 737-742, 2009.

Nemzetközi konferencia előadások

17. Tarczali, T., Süle, Z., Kalauz, K., Structural alternatives of business process models applying P-graph methodology, EDSI 2013, Budapest, Hungary, June 10-12, 2013.

16. Kalauz, K., Süle, Z., Bertók, B., Friedler, F., and Fan L. T., Extending process-network synthesis algorithms with time bounds for supply network design, PRES 2012, Praha, Czech Republic, August 25-29, 2012.

15. Szlama, A. G., Kalauz, K., Bertók, B., and Heckl, I., Solving a separation-network synthesis problem adopting interval optimization techniques, PRES 2012, Praha, Czech Republic, August 25-29, 2012.

14. Kalauz, K., Heckl, I., Borbás, P. I., Bertók, B., and L. Halász, L., Downstream Supply Chain Simulator and Planner, Factory Automation Conference 2012, Veszprém, Hungary, May 21-22, 2012.

13. Bertók, B., Kalauz, K., Süle, Z., Friedler, F., and Fan L. T., Algorithmic Synthesis of Process Networks wit Time Constraints by the P-garph Framework, VOCAL 2010, Veszprém, Hungary, December 13-15, 2010.

12. Kalauz, K., Süle, Z., Bertók, B., Friedler, F., and Fan L. T., Generation of Redundant Structures to Guarantee Predefined Level of Reliability in Business Processes, VOCAL 2010, Veszprém, Hungary, December 13-15, 2010.

11. Tarczali, T., Kadanoczki, H., Ősz, E., Zachár, G., Kalauz, K., and Süle, Z., P-graph Representation and Structural Analysis of Business Process Models, VOCAL 2010, Veszprém, Hungary, December 13-15, 2010.

10. Kalauz, K., Süle, Z., Bertók, B., Friedler, F., and Fan L. T., Reliability Analysis of Business Processes via the P-graph Framework, International Conference on Computational Management Science, Vienna, Austria, July 28-30, 2010.

9. Kalauz, K., and Bertók, B., Algorithmic Synthesis of Structural Alternatives for Business Processes, CSCS 2010, Szeged, Hungary, June 29-July 2, 2010.

8. Heckl, I., Kalocsai, P., Halász, L., and Kalauz, K., Event driven process simulation of pipeline networks, PRES 2009, Rome, Italy, May 10-13, 2009.

7. Adonyi, R., Kalauz, K., Bertók, B., Fan, L. T., and Friedler, F., Conceptual Design of Business Processes by Process-Network Synthesis, VOCAL 2008, Veszprém, Hungary, December 15-17, 2008.

6. Bertók, B., Kalauz, K., Friedler, F., and Fan, L. T., Reaction-Pathway Identification: Models and Algorithms, VOCAL 2008, Veszprém, Hungary, December 15-17, 2008.

5. Bertók, B., Kalauz, K., Friedler, F., Lin, Y.-C., Fan, L. T., and Shafie, S., Conceptual Design of Business Processes by Process-Network Synthesis, VOCAL 2008, Veszprém, Hungary, December 15-17, 2008.

4. Kalauz, K., Bertók, B., Fan, L. T., and F. Friedler, F., Reaction-pathway identification: methods and software, PRES 2008, Praha, Czech Republic, August 24-28, 2008.

3. Kalauz, K., Bertók, B., Friedler, F., and Fan L. T., Systematic Methods for Reaction-Pathway Identification, XIX Polish Conference of Chemical and Process Engineering, Rzeszow, Poland, September 3-7, 2007.

2. Bertók, B., Kalauz, K., and Fan L. T., Exhaustively Generating Basis Solutions for a System of Stoichiometrically Balanced Elementary Reactions by Resorting to the P-graph-based Method, VOCAL 2006, Veszprém, Hungary, December 13-15, 2006.

1. Kalauz, K., Bertók, B., Friedler, F., and Fan L. T., Reaction pathway identification: relations of methods based on graph theory and linear algebra, CHISA 2006, Praha, Czech Republic, August 27-31, 2006.

Hazai konferencia előadások

2. Kalauz, K., Süle, Z., Tarczali, T., Bertók, B., Friedler, F., and Fan L. T., P-gráf módszertan alkalmazása biztonságos üzleti folyamatok modellezésére, XXVIII. Magyar Operációkutatási Konferencia, Balatonőszöd, 2009.

1. Kalauz, K., Bertók, B., Friedler, F., and Fan L. T., Sztöchiometriai egyenletrendszerek minimális számú aktív változót tartalmazó megoldásainak generálása a P-gráf módszertan alkalmazásával, XXVII. Magyar Operációkutatási Konferencia, Balatonőszöd, 2007.

9. fejezet

Új tudományos eredmények

Az értekezés új tudományos eredményeinek tézisszerű összefoglalása:

1. Bevezettem a folyamat-hálózat szintézis feladat kölcsönös kizárási feltételekkel kibővített változatát.

   (a) Bizonyítottam, hogy a folyamat-hálózat szintézis algoritmusok az általam javasolt módosítások után, pontosan azokat a struktúrákat eredményezik, amelyek a folyamat-hálózatok szükséges kombinatorikus tulajdonságait kifejező axiómák mellett a kölcsönös kizárási feltételeket is teljesítik.

2. Bevezettem egy algoritmust, amely olyan módon bővíti a maximális struktúrát, hogy az abból generálható megoldásstruktúrák között az alternatív tevékenységek az összes lehetséges kombinációban megjelenhessenek. Redundancia esetén a kiegészítő tevékenységek garantálják, hogy az alternatív tevékenységek mindegyike olyan kapacitással szerepel az optimális struktúrában, hogy azok egymást kiválthassák.

   (a) Bevezettem az RG algoritmus, amely automatikusan építi be a redundanciákat a hálózat korlátai mentén. Továbbá úgy bővíti a kiindulási maximális struktúrát, hogy a folyamatszintézis algoritmusokat változtatás nélkül alkalmazva a korábbi megoldások mellett olyan hálózatokat is eredményeznek, amelyek egy vagy több tevékenység kiesése esetén is működőképesek maradnak.

(b) Körmentes hálózatok esetére megadtam egy olyan lineáris matematikai programozási modellt, amely megbízhatósági követelményeket teljesítő költségminimális hálózatokat szintetizáló korlátozás és szétválasztás algoritmusokban jól használható: mert kifejezi a megbízhatóság és a költség konkurenciáját, költségre és megbízhatóságra is alsó becslést ad, és kizárja a nem megvalósítható hálózatokat.

3. A PNS módszertant kibővítettem az erőforrások időbeli rendelkezésre állási, a végcélok teljesülési határidejének és a tevékenységek végrehajtási idejének figyelembe vételével.

   (a) Kidolgoztam egy algoritmust, amely a folyamat-hálózat maximális struktúrájában megjelenő alternatív tevékenységek esetén azok időbeli szinkronizálását kifejező kiegészítő tevékenységekkel bővíti a maximális struktúrát. Így lehetőséget adva a különböző megengedett struktúrákban az alternatív tevékenységek szinkron vagy aszinkron beépítésere.

   (b) Megmutattam, hogy a maximális struktúra kibővítése után a folyamatszintézis feladat megfelelő MILP formalizálásával a szinkron és az aszinkron működésű megoldások egyaránt megtalálhatóak matematikai programozással is.

   (c) Bebizonyítottam, hogy a maximális struktúrát kibővítő algoritmus futási ideje polinomiális.

Irodalomjegyzék

[1] P. A. Abdulla and R. Mayr, *Petri nets with time and cost*, In Infinity (2012), 9–24.

[2] R. Adonyi, J. Romero, L. Puigjaner, and F. Friedler, *Incorporating heat integration in batch process scheduling*, Applied Thermal Engineering **27** (2003), 1743–1762.

[3] H. Alla and R. David, *Continuous and hibrid petri nets*, In Proc. of the 8th European Workshop on Application and Theory of Petri Nets **8(1)** (1987), 275–294.

[4] H. Alla, R. David, and J. Le Bail, *Hybrid petri nets*, ECC91 European Control Conference, 1991, pp. 21–27.

[5] T. Allweyer, *Bpmn 2.0 (business process model and notation)*, BoD, 2010.

[6] M. Barany, B. Bertok, Z. Kovacs, F. Friedler, and L. T. Fan, *Optimization software for solving vehicle assignment problems to minimize cost and environmental impact of transportation*, Chemical Engineering Transactions **21** (2010), 499–504.

[7] ———, *Solving vehicle assignment problems by process-network synthesis to minimize cost and environmental impact of transportation*, Clean Technologies and Environmental Policy **13(4)** (2011), 637–642.

[8] H. P. Benson, *Separable concave minimization via partial outer approximation and branch and bound*, Oper. Res. Lett. **9** (1990), 389–394.

[9] Harold P. Benson and Reiner Horst, *A branch and bound–outer approximation algorithm for concave minimization over a convex set*, Comput. Math. Appl. **21** (1991), 67–76.

[10] B. Berard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux, *Comparison of different semantics for time petri nets*, Third international symposium on automated technology for verification and analysis, ATVA2005, LNCS, Springer **3707** (2005), 293–307.

[11] ———, *The expressive power of time petri nets*, Theoretical Computer Science **474** (2013), 1–20.

[12] B. Bertok, R. Adonyi, F. Friedler, and L. T. Fan, *Optimal design of supply chains by the p-graph framework*, ESCAPE 19, 2009.

[13] B. Bertok, K. Kalauz, Z. Sule, and F. Friedler, *Combinatorial algorithm for synthesizing redundant structures to increase reliability of supply chains: Application to biodisel suplly*, Industrial & Engineering Chemistry Research **52(1)** (2013), 181–186.

[14] Bertók Botond, *Folyamathálózatok strktúráinak algoritmikus szintézise*, Veszprém, 2003.

[15] M. Boyer and O. H. Roux, *On the compared expressiveness of arc, place and transition time petri nets*, Fundamenta Informaticae **88(2)** (2008), 225–249.

[16] Berliner BPM-Offensive, *Bpmn 2.0 poster*, `http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf`, 2011.

[17] G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario, *Timed state space analysis of real-time preemptive systems*, IEEE Transactions on Software Engineering **30(2)** (2004), 97–111.

[18] A. V. Cabot and S. S. Erenguc, *A branch and bound algorithm for solving a class of nonlinear integer programming problems*, Naval Res. Logist. Quart. **33** (1986), 559–567.

[19] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, Kluwer Academic Publishers, Boston, 1999.

[20] G. Chiola, M. A. Marsan, G. Balbo, and G. Conte, *Generalized stochastic petri nets: A definition at the net level and its implications*, IEEE Transactions on Software Engineering **19** (1993), 89–107.

[21] M. M. Daichendt and I. E Grossmann, *Preliminary screening procedure for the minlp synthesis of process systems I. aggregation and decomposition*, Comp. Chem. Eng. **18** (1994), 663–677.

[22] F. Daniel, *Business process management workshops*, Springer, 2012.

[23] R. David and H. Alla, *Continuous petri nets*, Journal of Circuits, Systems and Computers, Special Issue on Petri Nets (1998), 159–188.

[24] ———, *On hybrid petri nets*, Discrete Event Dynamic Systems **11(1-2)** (2001), 9–40.

[25] ———, *Discrete, continuous and hybrid petri nets (revised 2nd edition)*, Springer, Berlin, 2010.

[26] J.L. Devore, *Probability and statistics for engineering and the sciences*, Brooks/Cole Pub. Co., Pacific Grove, 1991.

[27] A. DiFebbraro, D. Giglio, and N. Sacco, *Urban traffic control structure based on hybrid petri nets*, IEEE Transactions on Intelligent Transportation Systems **5(4)** (2004), 224–237.

[28] V. Dua and E. N. Pistikopoulos, *An outer-approximation algorithm for the solution of multiparametric MINLP problems*, Comp. Chem. Eng. **22** (1998), S955–S958.

[29] M. Dumas, M. La Rosa, J. Mendling, and H.A. Reijers, *Fundamentals of business process management*, Springer, 2013.

[30] M. A. Duran and I. E. Grossmann, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Math. Programming **36** (1986), 307–339.

[31] A. Elkamel, *Scheduling of process operations using mathematical programming techniques*, PhD Thesis, Purdue University (1993).

[32] Bartha T. és Csertán Gy. és Majzik I. Pataricza A. és Varró D. és Varró-Gyapay Sz., *Formális módszerek az informatikában*, Typotex, Budapest, 2004.

[33] Cs. Fabian, *Linx, an interactive linear programming library*, 1992.

[34] J. E. Falk and R. M. Soland, *An algorithm for separable nonconvex programming problems*, Management Science **15** (1969), 550–569.

[35] L. T. Fan, B. Bertok, F. Friedler, and S. Shafie, *Mechanisms of ammonia-synthesis reaction revisited with the aid of a novel graph-theoretic method for determining candidate mechanisms in deriving the rate law of a catalytic reaction*, Hung. J. Ind. Chem. **29** (2001), 71–80.

[36] W. Feller, *An introduction to probability theory and its applications*, Wiley, New York, 1994.

[37] G. Feng and L. T. Fan, *On stream splitting in separation system sequencing*, Industrial and Engineering Chemistry Research **35(6)** (1996), 1951–1958.

[38] C. A. Floudas and X. Lin, *Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review*, Computers and Chemical Engineering **28** (2004), 2109–2129.

[39] C. A. Floudas and V. Visweswaran, *A global optimization algorithm (GOP) for certain classes of nonconvex NLPs - I. Theory*, Comp. Chem. Eng. **14** (1990), 1397–1417.

[40] E. Fraca, J. Julvez, and M. Silva, *Marking homothetic monotonicity and fluidization of untimed petri nets*, In International workshop on discrete event systems, WODES12 (2012), 21–27.

[41] E. S. Fraga and K. I. M. McKinnon, *The use of dynamic programming with parallel computers for process synthesis*, Computers chem. Engng **18** (1994), 1–13.

[42] J. Freund and B. Rücker, *Real-life bpmn: Using bpmn 2.0 to analyze, improve, and automate processes in your company*, CreateSpace, 2012.

[43] F. Friedler, L. T. Fan, and B. Imreh, *Process network synthesis: Problem definition*, Networks **31** (1998), 119–124.

[44] F. Friedler, K. Tarjan, Y. W. Huang, and L. T. Fan, *Combinatorial algorithms for process synthesis*, Comp. Chem. Eng. **16** (1992), S113–S320.

[45] ———, *Graph-theoretic approach to process synthesis: Axioms and theorems*, Chem. Eng. Sci. **47** (1992), 1973–1988.

[46] ———, *Graph-theoretical approach to process synthesis: Polynomial algorithm for maximal structure generation*, Comp. Chem. Eng. **17** (1993), 929–942.

[47] F. Friedler, J. B. Varga, and L. T. Fan, *Decision-mapping: A tool for consistent and complete decisions in process synthesis*, Chem. Eng. Sci. **50** (1995), 1755–1768.

[48] F. Friedler, J. B. Varga, E. Feher, and L. T. Fan, *Combinatorial 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), pp. 609–626, Kluwer Academic Publishers, Norwell, MA, U.S.A., 1996.

[49] K. Garimella, M. Lees, and B. Williams, *Bpm basics for dummies, software ag special edition*, Wiley Publishing, Inc., Hoboken, NJ, 2008.

[50] F. Glover, *Improved linear integer programming formulations of nonlinear integer problems*, Man. Sci. **22(4)** (1975), 455–460.

[51] BV Gnedenko and I.A. Ushakov, *Probabilistic reliability engineering*, Wiley, New York, 1995.

[52] J. Gouws and T. Majozi, *Usage of inherent storage for minimization of wastewater in multipurpose batch plants*, Chemical Engineering Science **64** (2009), 3545–3554.

[53] Sz. Gyapay and A. Pataricza, *A combination of petri nets and process network synthesis*, Proceedings of the IEEE International Conference on Systems, Man & Cybernetics (2003), 1167–1174.

[54] J. Happel and P. H. Sellers, *Analysis of the possible mechanisms for a catalytic reaction system*, Adv. Catal. **32** (1983), 273–323.

[55] J. Happel, P. H. Sellers, and M. Otarod, *The characterization of complex systems of chemical reactions*, Ind. Eng. Chem. Res. **29** (1990), 1057–1064.

[56] A. Hoyland and M. Rausand, *System reliability theory: Models and statistical methods*, Wiley, New York, 1994.

[57] T. Ibaraki, *Theoretical comparisons of search strategies in branch-and-bound algorithms*, Journal of Computer and Information Science **5** (1976), 315–344.

[58] M. G. Ierapetritou and C. A. Floudas, *Effective continuous-time formulation for short-term scheduling. 1. multipurpose batch processes*, Ind. Eng. Chem. Res. **37** (1998), 4341–4359.

[59] B. Imreh, F. Friedler, and L. T. Fan, *An algorithm for improving the bounding procedure in solving process network synthesis by a branch-and-bound method*, Nonconvex Optimization and its Applications (Eds: C. A. Floudas and P. M. Pardalos), pp. 315–348, Kluwer Academic Publishers, Dordrecht, 1997.

[60] K. Jensen, *Coloured petri nets and the invariant-method*, Theoretical Computer Science **14** (1981), 317–336.

[61] J. Julvez, C. Mahulea, and C. Vazquez, *Simhpn: A matlab toolbox for simulation, analysis and design with hybrid petri nets*, Nonlinear Analysis: Hybrid Systems **6(2)** (2012), 806–817.

[62] K. Kalauz and B. Bertok, *Algorithmic synthesis of structural alternatives for business processes*, CSCS 2010, June 29 - July 02 2010.

[63] K. Kalauz, Z. Sule, B. Bertok, F. Friedler, and Fan L. T., *Generation of redundant structures to guarantee predefined level of reliability in business processes*, VOCAL 2010, December 13-15 2010.

[64] ———, *Reliability analysis of business processes via the p-graph framework*, International Conference on Computational Management Science, July 28-30 2010.

[65] K. Kalauz, Z. Süle, T. Tarczali, B. Bertok, F. Friedler, and Fan L. T., *P-gráf módszertan alkalmazása biztonságos üzleti folyamatok modellezésére*, XXVIII: Magyar Operációkutatási Konferencia, jĂşnius 8-10 2009.

[66] E. Kondili, C. Pantelides, C, and R. W. H. Sargent, *A general algorithm for short-term scheduling of batch operations-i. milp formulation*, Comp. Chem. Eng. **17(2)** (1993), 211–227.

[67] Z. Kovacs, L. T. Fan, and F. Friedler, *Folyamathalozatok megbizhatosaga*, XXX. Magyar Operaciokutatasi Konferencia, 2013.

[68] Takahito Kuno and Takahiro Utsunomiya, *A Lagrangian based branch-and-bound algorithm for production-transportation problems*, J. Global Optim. **18** (2000), 59–73.

[69] E. L. Lawler and D. E. Wood, *Branch-and-bound methods: A survey*, Operations Research **14** (1966), 699–719.

[70] D. Liu, J. Wang, S. C. F. Chan, J. Sun, and L. Zhang, *Modeling workflow processes with colored petri nets*, Computers in Industry **49(3)** (2002), 267–281.

[71] D. G. Luenberger, *Intoduction to linear and nonlinear programming*, Addison-Wesley Publishing Co., Massachusetts, 1973.

[72] A. M. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with generalized stochastic petri nets*, Wiley, Bognor Regis, England, 1995.

[73] M. A. Marsan, G. Balbo, and G. Conte, *A class of generalized stochastic petri nets for the performance analysis of multiprocessor systems*, ACM Transactions on Computer Systems **2(2)** (1984), 93–122.

[74] M. A. Marsan, A. Bobbio, and S. Donatelli, *Petri nets in performance analysis: An introduction*, In Lectures on Petri Nets I: Basic Models (1998), 211–256.

[75] L. Mockus and G. V. Reklaitis, *Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization*, Comp. Chem. Eng. **21** (1997), 1147–1156.

[76] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, John Wiley & Sons, New York, 1988.

[77] Object Management Group (OMG), *Business process model and notation (bpmn)*, `http://www.omg.org/spec/BPMN/2.0/PDF/`, Accessed January 3, 2011.

[78] T. Panagacos, *The ultimate guide to business process management: Everything you need to know and how to apply it to your organization*, CreateSpace, 2012.

[79] S. Panek, S. Engell, S. Subunatarajan, and O. Stursberg, *Scheduling of multiproduct batch plants based upon timed automata models*, Computers and Chemical Engineering **32** (2008), 275–291.

[80] C. C. Pantelides, *Unified frameworks for optimal process planning and scheduling*, Proc. Second Conference on Foundations of Computer-Aided Operations (FOCAPO II) (1994), 235–274.

[81] W. Penczek and A. Polrola, *Advances in verification of time petri nets and timed automata, a temporal logic approach*, Springer Berlin Heidelberg, Berlin, 2006.

[82] J. L. Peterson, *Petri net theory and the modeling of systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[83] C. A. Petri, *Kommunikation mit automaten*, PhD Dissertation, Universität Bonn (1962).

[84] ———, *Communication with automata*, Rome Air Development Center-TR-65-377, New York, 1966.

[85] H. Pham, *System software reliability*, Springer, London, 2007.

[86] A. T. Phillips and J. B. Rosen, *Sufficient conditions for solving linearly constrained separable concave global minimization problems*, J. Global Optim. **3** (1993), 79–74.

[87] R. Raman and I. E. Grossmann, *Symbolic integration of logic in MILP branch and bound methods for the synthesis of process networks*, Annals of Operations Research **42** (1993), 169–191.

[88] C. Ramchandani, *Analysis of asynchronous concurrent systems by timed petri nets*, Massachusetts Institute of Technology Cambridge, Cambridge, MA, USA, 1974.

[89] L. Recalde, S. Haddad, and M. Silva, *Continuous petri nets: Expressive power and decidability issues*, Int. Journal of Foundations of Computer Science **21(2)** (2010), 235–256.

[90] L. Recalde and M. Silva, *Petri nets fluidification revisited: Semantics and steady state*, European Journal of Automation APII-JESA **35(4)** (2001), 435–449.

[91] J. Romero, A. Espuna, F. Friedler, and L. Puigjaner, *A new framework for batch process optimization using the flexible recipe*, Industrial and Engineering Chemistry Research **42(2)** (2003), 370–379.

[92] N. V. Sahinidis, I. E. Grossmann, R. E. Fornari, and M. Chathrathi, *Optimisation model for long-range planning in the chemical industry*, Comp. Chem. Eng. **15** (1991), 255–272.

[93] E. Sanmarti, F. Friedler, and L. Puigjaner, *Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation*, Comp. Chem. Eng. **22** (1998), 847–850.

[94] E. Sanmarti, L. Puigjaner, T. Holczinger, and F. Friedler, *Combinatorila framwork for effective scheduling of multipurpose batch plants*, Process System Engineering **48** (2002), 2557–2570.

[95] G. Schilling and C. C. Pantelides, *A simple continuous time process scheduling formulation and a novel solution algorithm*, Comp. Chem. Eng. **S20** (1996), S1221–S1226.

[96] N. Shah, C. C. Pantelides, and R. W. H. Sargent, *A general algorithm for short-term scheduling of batch operations - 2. computational issues*, Comp. Chem. Eng. **17** (1993), 229–244.

[97] K. J. Sherry, *Bpmn pocket reference: A practical guide to the international business process model and notation standard bpmn version 2.0*, CreateSpace, 2010.

[98] J. J. Siirola, *Industrial applications of chemical process synthesis*, Advances in Chemical Engineering **23** (1996), 1–62.

[99] M. Silva, J. Julvez, C. Mahulea, and C. Vazquez, *On fluidization of discrete event models: Observation and control of continuous petri nets*, Discrete Event Dynamic Systems **21** (2011), 427–497.

[100] M. Silva and L. Recalde, *Petri nets and integrality relaxations: A view of continuous petrinets*, IEEE Trans. on Systems, Man, and Cybernetics **32(4)** (2002), 314–327.

[101] ———, *On fluidification of petri nets: from discrete to hybrid and continuous models*, Annual Reviews in Control **28(2)** (2004), 253–266.

[102] M. Silva and M. Recalde, *On fluidification of petri net models: from discrete to hybrid and continuous models*, Annual Reviews in Control **28** (2004), 253–266.

[103] E. Sule, *Az időszerepe és jelentősége az ellátási láncban*, Győr, 2010.

[104] T. Tarczali, Kadanoczki. H., E. Osz, G. Zachar, K. Kalauz, and Z. Sule, *P-graph representation and structural analysis of business process models*, VOCAL 2010, December 13-15 2010.

[105] M. I. Temkin, *The kinetics of steady-state complex reactions*, Int. Chem. Eng. **11** (1971), 709–717.

[106] W. M. P. van der Aalst, *Interval timed coloured petri nets and their analysis*, Application and Theory of Petri Nets **691** (1993), 453–472.

[107] C. R. Vazquez, C. Mahulea, J. Julvez, and M. Silva, *Control of discrete-event systems automata and petri net perspectives*, Springer London, London, 2013.

[108] V. Visweswaran and C. A. Floudas, *A global optimization algorithm (GOP) for certain classes of nonconvex NLPs - II. application of theory and test problems*, **14** (1990), 1419–1434.

[109] J. Wang, *Timed petri nets: Theory and application*, Kluwer Academic Publishers, Norwell, 1998.

[110] M. Weske, *Business process management: Concepts, languages, architectures (second edition)*, Springer, Berlin, 2012.

[111] K. L. Yee and N. Shah, *Improving the efficiency of discrete-time scheduling formulations*, Comp. Chem. Eng. **S22** (1998), S403–S410.

[112] X. Yuan, L. Pibouleau, and Domenech S., *Experiments in process synthesis via mixed-integer programming*, Chem. Eng. Process. **25** (1989), 99–116.

[113] X. Zhang, *Algorithms for optimal scheduling using nonlinear models*, PhD Thesis, University of Londona (1995).

[114] X. Zhang and R. W. H. Sargent, *The optimal operation of mixed production facilities - a general formulation and some solution appoaches for the solution*, Proc. of 5th Intl. Symp. on Process Systems Engineering, Kyongju, Korea (1994), 171–177.

Tárgymutató

Állapot-tevékenység hálózat (STN), 15–18
Általánosított sztochasztikus Petri háló, 11–12
Árazott és időzített Petri háló, 12
Árazott és időzített Petri háló, 12

A PNS és a Petri háló összehasonlítás, 13–15
A relaxált feladat, 41
ABB algoritmus, 39–40, 44–47
ABBMX algoritmus, 57–62
ABBMX függvény, 57

BPD, 5
BPM, 5
BPMN, 5–7
BranchABBMX függvény, 58
BranchSSGMX függvény, 52

Determinisztikus időzített Petri háló, 11
Determinisztikus időzített Petri hálók, 11

Elérhetőségi probléma, 13
Erőforrás-tevékenység hálózat (RTN), 18
Erőforrás-tevékenység hálózatot (RTN), 20

Folyamathálózat-szintézis (PNS), 24–29
Folytonos Petri háló, 10

Hibrid Petri háló, 10

Időkorlátos PNS (TCPNS), 110–111
Időzített Petri háló, 10–11

Megbízhatóság a P-gráf módszertanban, 85–88
MSETC algoritmus, 117
MSG algoritmus, 35
MX függvény, 61

P-gráf, 29–32
Petri háló, 7–15

Redundancia generálás, 81–85
Relaxált feladat, 42
RG algoritmus, 82

S-gráf, 20–24
S1, S2, ..., S5 axiómák, 34–35
SolveRelaxedLPMX függvény, 58
SSG algoritmus, 35–39
SSGMX algoritmus, 52–56
SSGMX függvény, 52
Strukturális leképezések, 32–34
Színezett Petri háló, 12–13

Mellékletek

A most következő részben a 6. fejezetben bemutatott időorlátos folyamathálózatok szintézisére kidolgozott TCPNS bemutatásra használt két szemléltető példáknak az IBM ILOG CPLEX Optimization Studioval (Version 12.5) történő megvalósítását mutatom be. A 1. Mellékletben az általam kidolgozott TCPNS megoldó módszer, a 2. Mellékletben a 6.3. fejezetben bemutatott jármű hozzárendelési probléma, a 3. Mellékletben pedig a 6.5. fejezetben ismertetett előidejűség kezelésre bemutatott példa IBM ILOG CPLEX Optimization Studioval történő megvalósítása látható.

1. Melléklet: A TCPNS megoldó módszer megvalósítása IBM ILOG CPLEX Optimization Studioval

A 6. fejezetben bemutatott TCPNS megoldó módszert IBM ILOG CPLEX környezetben két fájllal valósítjuk meg. Elsőként tekintsük a *tcpns.mod* fájlt, amely tartalmazza magát megoldó módszert.

```
/*****************************************
* tcpns.mod
* OPL 12.5 Model
* Author: Karoly Kalauz
*****************************************/

include "tcpns_types.mod";

using CPLEX;

dvar int decision[opUnit] in 0..1;
dvar int+ opcap[opUnit];
dvar int+ optime[opUnit];
dvar int+ mattime[material];

execute
{
    writeln("PNS FELADAT");
    writeln("");
    writeln("Raw material(s):");
    for(var mat in rawMat)
        writeln("id: " + mat.mid.mid + "\t\t nev: " + mat.mid.name +
        "\t\t tipus: " + mat.mid.type + "\t\t LB: " + mat.mid.lowerBound +
```

```
        "\t\t UB: " + mat.mid.upperBound + "\t\t cost: " + mat.cost +
        "\t\t availability time: " + mat.mid.availabilityTime);

    writeln("");
    writeln("Intermedite material(s):");
    for(mat in intermediateMat)
        writeln("id: " + mat.mid.mid + "\t\t nev: " + mat.mid.name +
        "\t\t tipus: " + mat.mid.type + "\t\t LB: " + mat.mid.lowerBound +
        "\t\t UB: " + mat.mid.upperBound + "\t\t price: " + mat.price +
        "\t\t penalty: " + mat.penalty);

    writeln("");
    writeln("Product(s):");
    for(mat in product)
        writeln("id: " + mat.mid.mid + "\t\t nev: " + mat.mid.name +
        "\t\t tipus: " + mat.mid.type + "\t\t LB: " + mat.mid.lowerBound +
        "\t\t UB: " + mat.mid.upperBound + "\t\t price: " + mat.price +
        "\t\t deadline: " + mat.mid.deadline);

    writeln("");
    writeln("Operating Unit(s):");
    for(var op in opUnit)
        writeln("id: " + op.oid + "\t\t nev: " + op.name +
        "\t\t capacity: " + op.capacity + "\t\t fix cost: " + op.fixCost +
        "\t\t prop cost: " + op.propCost + "\t\t fix time: " + op.fixTime +
        "\t\t prop time: " + op.propTime);

    writeln("");
    writeln("Material_to_operating_unit_flow_rates:");
    for(op in opUnit)
    {
```

```
write(op.name + ": ");
var db = 0;
for(var mo in relationMatToUnit)
{
    if (mo.oid == op)
    {
        if (db==0)
        {
            write(mo.rate + " " + mo.mid.name);
            db = db + 1;
        }
        else
            write(" + " + mo.rate + " " + mo.mid.name);
    }
}
write(" => ");
db = 0;
for(mo in relationMatFromUnit)
{
    if (mo.oid == op)
    {
        if (db==0)
        {
            write(mo.rate + " " + mo.mid.name);
            db = db + 1;
        }
        else
            write(" + " + mo.rate + " " + mo.mid.name);
    }
}
writeln(" ");
```

```
    }
}

minimize
    sum(mat in rawMat)
        (
            mat.cost *
                ( sum (mo1 in relationMatToUnit : mo1.mid == mat.mid)
                (opcap[mo1.oid] * mo1.rate))
        )
    + sum(op in opUnit) (op.fixCost * decision[op] + op.propCost * opcap[op]);

subject to
{
    // Korlátozási feltétel tevékenységekre:
    forall(op in opUnit)
        opcap[op] <= op.capacity * decision[op];

    forall(op in opUnit)
        opcap[op] >= 0.0001 * decision[op];

    // Korlátozási feltétel erőforrásokra:
    forall(mat in rawMat)
        sum (mo in relationMatToUnit : mo.mid == mat.mid) opcap[mo.oid] *
            mo.rate <= mat.mid.upperBound;

    // Korlátozási feltétel köztes entitásokra:
    forall(mat in intermediateMat)
    {
        sum (mo1 in relationMatFromUnit : mo1.mid == mat.mid) opcap[mo1.oid] *
            mo1.rate - sum (mo2 in relationMatToUnit : mo2.mid == mat.mid) opcap[mo2.o
```

```
        mo2.rate >= mat.mid.lowerBound;
    sum (mo1 in relationMatFromUnit : mo1.mid == mat.mid) opcap[mo1.oid] *
        mo1.rate - sum (mo2 in relationMatToUnit : mo2.mid == mat.mid) opcap[mo2.o
        mo2.rate <= mat.mid.upperBound;
}

// Korlátozási feltétel a végcélokra:
forall(mat in product)
{
    sum (mo1 in relationMatFromUnit : mo1.mid == mat.mid) opcap[mo1.oid] *
        mo1.rate - sum (mo2 in relationMatToUnit : mo2.mid == mat.mid) opcap[mo2.o
        mo2.rate >= mat.mid.lowerBound;
    sum (mo1 in relationMatFromUnit : mo1.mid == mat.mid) opcap[mo1.oid] *
        mo1.rate - sum (mo2 in relationMatToUnit : mo2.mid == mat.mid) opcap[mo2.o
        mo2.rate <= mat.mid.upperBound;
}

// Időkorlát az entitásokra
forall(mat in material)
{
    mattime[mat] >= mat.availabilityTime;
    mattime[mat] <= mat.deadline;
}

forall(op in opUnit)
{
    forall(mo in relationMatToUnit : mo.oid == op)
        optime[op] >= mattime[mo.mid];
}

forall(op in opUnit)
```

```
    {
        forall(mo in relationMatFromUnit : mo.oid == op)
            mattime[mo.mid] >= optime[op] + decision[op]*op.fixTime +
                op.propTime * opcap[op] + decision[op]*300-300;
    }
}

execute
{
    writeln("");
    writeln("OPTIMAL SOLUTION:");
    writeln("");
    writeln("Operating units in operation:");
    for(var op in opUnit)
    {
        if (decision[op] == 1)
        {
            writeln(op.name + "\t capacity: " + opcap[op] + "\t WorksFrom: " + optime[op
            result.add(op.oid, opcap[op], optime[op]);
        }
    }
    for(var mat in material)
    {
        writeln(mat.name + "\t time: " + mattime[mat]);
        materialTime.add(mat.mid, mattime[mat]);
    }
    writeln(result);
}
```

Másodikként pedig tekintsük a *tcpns_ types.mod* fájlt, amely az egyes adatszerkezetek definícióit tartalmazza.

```
/*****************************************
* tcpns_types.mod
* OPL 12.5 Model
* Author: Karoly Kalauz
*****************************************/

tuple TMaterial
{
    key int mid;
    string name;
    string type;
    float lowerBound;
    float upperBound;
    int availabilityTime;
    int deadline;
};

tuple TOpUnit
{
    key int oid;
    string name;
    float capacity;
    float fixCost;
    float propCost;
    int fixTime;
    int propTime;
};
{TMaterial} material = ...;
```

```
{TOpUnit} opUnit = ...;

tuple TIntermediateMat
{
    key TMaterial mid;
    float price;
    float penalty;
};

tuple TProduct
{
    key TMaterial mid;
    float price;
};

tuple TRawMat
{
    key TMaterial mid;
    float cost;
};

tuple TRelationMatFromUnit
{
    key TOpUnit oid;
    key TMaterial mid;
    float rate;
};

tuple TRelationMatToUnit
{
    key TOpUnit oid;
```

```
    key TMaterial mid;
    float rate;
};

{TIntermediateMat} intermediateMat with mid in material = ...;
{TProduct} product with mid in material = ...;
{TRawMat} rawMat with mid in material = ...;
{TRelationMatFromUnit} relationMatFromUnit with oid in opUnit, mid in material = ...;
{TRelationMatToUnit} relationMatToUnit with oid in opUnit, mid in material = ...;

// Output
tuple TMaterialTime
{
    key int mid;
    int availableFrom;
};
tuple TResult
{
    key int oid;
    float capacity;
    int worksFrom;
};

{TMaterialTime} materialTime;
{TResult} result;
```

2. Melléklet: A dolgozatban bemutatott jármű hozzárendelési feladat megvalósítása IBM ILOG CPLEX Optimization Studioval

A most következő részben a 6.3. fejezetben látható jármű hozzárendelési feladat IBM ILOG CPLEX Optimization Studioval történő megvalósítását mutatom be.A feladat megoldásához az 1. Mellékletben bemutatott *tcpns.mod* fájl és a *tcpns_types.mod* alap fájlok mellé szükség van még az adott feladat pontos paramétereinek a megadására. Ezt a következőekben bemutatásra kerülő *tcpns_fuvarszervezes.dat* fájlal adhatjuk meg.

```
/*******************************************
* tcpns_fuvarszervezes.dat
* OPL 12.5 Data
* Author: Karoly Kalauz
*******************************************/

material =
{
    <1 "r3" "Raw" 0 1 0 10000>
    <2 "r2" "Raw" 0 1 0 10000>
    <3 "r1" "Raw" 0 1 10 10000>
    <4 "eur" "Raw" 0 1000000 0 10000>
    <5 "co2" "Raw" 0 1000000 0 10000>
    <6 "r3_veszprem" "Intermediate" 0 1000000 0 10000>
    <7 "r2_veszprem" "Intermediate" 0 1000000 0 10000>
    <8 "r2_tihany" "Intermediate" 0 1000000 0 10000>
    <9 "r2_almadi" "Intermediate" 0 1000000 0 10000>
    <10 "r1_almadi" "Intermediate" 0 1000000 0 10000>
```

```
    <11 "r1_veszprem" "Intermediate" 0 1000000 0 10000>
    <12 "r1_fehervar" "Intermediate" 0 1000000 0 10000>
    <13 "r1_tihany" "Intermediate" 0 1000000 0 10000>
    <14 "km_r3" "Intermediate" 0 1000000 0 10000>
    <15 "km_r2" "Intermediate" 0 1000000 0 10000>
    <16 "km_r1" "Intermediate" 0 1000000 0 10000>
    <17 "r1_veszprem2" "Intermediate" 0 1000000 0 10000>
    <18 "r1_tihany2" "Intermediate" 0 1000000 0 10000>
    <19 "p3_veszprem_tihany" "Product" 1 1 0 130>
    <20 "p2_veszprem_fehervar" "Product" 1 1 0 160>
    <21 "p1_tihany_almadi" "Product" 1 1 0 180>
};

opUnit =
{
    <1 "r3_start_kenese_veszprem" 1 0 0 60 0>
    <2 "r3_p2_vp_szf" 1 0 0 80 0>
    <3 "r2_start_fehervar_veszprem" 1 0 0 80 0>
    <4 "r2_start_fehervar_tihany" 1 0 0 140 0>
    <5 "r2_p3_veszprem_tihany" 1 0 0 60 0>
    <6 "r2_p1_tihany_almadi" 1 0 0 40 0>
    <7 "r2_almadi_vp" 1 0 0 50 0>
    <8 "r1_p1_tihany2_almadi" 1 0 0 40 0>
    <9 "r1_start_tihany_veszprem2" 1 0 0 60 0>
    <10 "r1_almadi_veszprem2" 1 0 0 50 0>
    <11 "r1_p2_veszprem_fehervar" 1 0 0 80 0>
    <12 "r1_p3_veszprem2_tihany" 1 0 0 60 0>
    <13 "r1_tihany_veszprem" 1 0 0 60 0>
    <14 "r1_fehervar_veszprem2" 1 0 0 80 0>
    <15 "r1_fehervar_tihany2" 1 0 0 140 0>
    <16 "r1_tihany_tihany2" 1 0 0 0 0>
```

```
    <17 "travel_r3" 10000 0 0 0 0>
    <18 "travel_r2" 10000 0 0 0 0>
    <19 "travel_r1" 10000 0 0 0 0>
    <20 "r1_start_tihany_veszprem" 1 0 0 60 0>
    <21 "r1_almadi_veszprem" 1 0 0 50 0>
    <22 "r1_start_tihany2" 1 0 0 0 0>
};

intermediateMat =
{
    <<6> 0 0>
    <<7> 0 0>
    <<8> 0 0>
    <<9> 0 0>
    <<10> 0 0>
    <<11> 0 0>
    <<12> 0 0>
    <<13> 0 0>
    <<14> 0 0>
    <<15> 0 0>
    <<16> 0 0>
    <<17> 0 0>
    <<18> 0 0>
};

product =
{
    <<19> 0>
    <<20> 0>
    <<21> 0>
};
```

```
rawMat =
{
    <<1> 0>
    <<2> 0>
    <<3> 0>
    <<4> 1>
    <<5> 0>
};

relationMatFromUnit =
{
    <<1> <6> 1>
    <<2> <20> 1>
    <<3> <7> 1>
    <<4> <8> 1>
    <<5> <8> 1>
    <<5> <19> 1>
    <<6> <9> 1>
    <<6> <21> 1>
    <<7> <7> 1>
    <<8> <10> 1>
    <<8> <21> 1>
    <<9> <17> 1>
    <<10> <11> 1>
    <<11> <12> 1>
    <<11> <20> 1>
    <<12> <13> 1>
    <<12> <19> 1>
    <<13> <11> 1>
    <<14> <17> 1>
```

```
    <<15> <18> 1>
    <<16> <18> 1>
    <<17> <14> 1>
    <<18> <15> 1>
    <<19> <16> 1>
    <<20> <11> 1>
    <<21> <17> 1>
    <<22> <18> 1>
};

relationMatToUnit =
{
    <<1> <1> 1>
    <<1> <14> 60>
    <<2> <6> 1>
    <<2> <14> 80>
    <<3> <2> 1>
    <<3> <15> 80>
    <<4> <2> 1>
    <<4> <15> 140>
    <<5> <7> 1>
    <<5> <15> 60>
    <<6> <8> 1>
    <<6> <15> 40>
    <<7> <9> 1>
    <<7> <15> 50>
    <<8> <16> 40>
    <<8> <18> 1>
    <<9> <3> 1>
    <<9> <16> 60>
    <<10> <10> 1>
```

```
    <<10> <16> 50>
    <<11> <16> 80>
    <<11> <11> 1>
    <<12> <16> 60>
    <<12> <17> 1>
    <<13> <13> 1>
    <<13> <16> 60>
    <<14> <12> 1>
    <<14> <16> 80>
    <<15> <12> 1>
    <<15> <16> 140>
    <<16> <13> 1>
    <<16> <16> 1>
    <<17> <4> 0.4>
    <<17> <5> 300>
    <<18> <4> 0.5>
    <<18> <5> 400>
    <<19> <4> 0.6>
    <<19> <5> 375>
    <<20> <3> 1>
    <<20> <16> 60>
    <<21> <10> 1>
    <<21> <16> 50>
    <<22> <3> 1>
    <<22> <16> 1>
};
```

3. Melléklet: A dolgozatban bemutatott előidejűség kezelését bemutató feladat megvalósítása IBM ILOG CPLEX Optimization Studioval

A most következő részben a 6.5. fejezetben látható előidejűség kezelését szemléltető feladat IBM ILOG CPLEX Optimization Studioval történő megvalósítását mutatom be. A feladat megoldásához az 1. Mellékletben bemutatott *tcpns.mod* és a *tcpns_types.mod* alap fájlok mellé szükség van még az adott feladat pontos paramétereinek a megadására. Ezt a következőekben bemutatásra kerülő *tcpns_eloideju.dat* fájlal adhatjuk meg.

```
/*****************************************
* tcpns_eloideju.dat
* OPL 12.5 Data
* Author: Karoly Kalauz
*****************************************/

material =
{
    <1 "m1" "Raw" 0 10 0 10000>
    <2 "m2" "Raw" 0 10 1 10000>
    <3 "m3" "Intermediate" 0 1000 0 10000>
    <4 "m4" "Intermediate" 0 1000 0 10000>
    <5 "m5" "Product" 2 1000 0 13>
    <6 "m6" "Product" 1 1000 0 14>
    <7 "mt1" "Intermediate" 0 1000 0 10000>
    <8 "mt2" "Intermediate" 0 1000 0 10000>
};

product =
```

```
{
    <<5> 0>
    <<6> 0>
};

rawMat =
{
    <<1> 1>
    <<2> 2>
};

opUnit =
{
    <1 "O1" 10 1 1 8 0>
    <2 "O2" 10 2 1 3 1>
    <3 "O3" 1000000 3 1 4 0>
    <4 "O4" 1000000 4 1 9 0>
    <5 "Ot1" 1000000 0 0 0 0>
    <6 "Ot2" 1000000 0 0 0 0>
};

intermediateMat =
{
    <<3> 0 0>
    <<4> 0 0>
    <<7> 0 0>
    <<8> 0 0>
};

relationMatFromUnit =
{
```

```
    <<1> <7> 1>
    <<1> <8> 1>
    <<2> <4> 1>
    <<3> <5> 1>
    <<4> <6> 1>
    <<5> <3> 1>
    <<6> <4> 1>
};

relationMatToUnit =
{
    <<1> <1> 1>
    <<2> <2> 1>
    <<3> <3> 1>
    <<4> <4> 1>
    <<5> <7> 1>
    <<6> <8> 1>
};
```