

Ruppert Tamás

FOLYAMATBÁNYÁSZATI ALGORITMUSOK

FEJLESZTÉSE

**Témavezető: Dr. Abonyi János, Dr. Király
András**

Mérnöki Kar, Folyamatmérnöki Intézeti Tanszék

Országos Tudományos Diákköri Konferencia

2015



Tartalom

Kivonat.....	1
Abstract	2
1 Célkitűzés és irodalmi áttekintés	3
2 Algoritmus fejlesztés specifikációi	7
2.1 A keretrendszer szoftvereinek megválasztása	7
2.2 Szoftverrendszer felépítése.....	8
3 A folyamatbányászat elméleti eszköztára.....	9
3.1 Alpha algoritmus	14
3.2 Heurisztikus bányászat	18
3.3 Folyamatlépések csoportosítása	23
4 Folyamatmodellek vizualizációja	28
4.1 Él gyakoriságok megjelenítése	30
4.2 Folyamatok klaszterezésének megjelenítése	30
4.3 Súlyozások vizualizációja	31
5 Alpha algoritmus fejlesztése	33
6 Fejlesztett súlyozási módszerek.....	35
6.1 Súlyozás a heurisztikus folyamatbányászatban.....	35
6.2 Súlyozás a folyamatlépések csoportosításában	37
7 Esettanulmányok.....	38
7.1 Alkalmazás logisztikai folyamat elemzésében	38
7.2 Alkalmazás pénzügyi kockázatok feltárásában	42
7.3 Tanulmányi előrehaladást leíró folyamatmodell identifikálása	44
8 Összefoglalás	49
9 Irodalomjegyzék	51

Kivonat

A termelő és üzleti folyamatok fejlesztése és optimalása megfelelő részletességű modelleket igényel. A termeléstámogató és a vállalatirányítási információs rendszerek úgynevezett eseménynaplókban rögzítik a folyamatlépések végrehajtásával kapcsolatos fontosabb információkat. Folyamatbányászati algoritmusok alkalmazásával lehetőség nyílik a folyamatmodellek ezen eseménynaplók alapján történő automatikus generálására és validálására. A modellek identifikációja az eseménynaplók és a folyamatok komplexitása, esetenként sztochasztikus jellege miatt nem triviális feladat.

A szakirodalomban fellelhető folyamatbányászati algoritmusok korlátozott módon, célirányos szoftverekben elérhetők, fejlesztésük a kevésbé nyitott rendszerek miatt korlátokba ütközik.

E korlátok feloldása érdekében egy széles körben elérhető folyamatbányászati eszköztárat hoztam létre az *Alpha*- és a *Heurisztikus* algoritmusok MATLAB implementációjával. Az algoritmusok elemzése során a futásidő, a számítási komplexitás és a skálázhatóság javítására vonatkozó fejlesztési lehetőségeket foglalmaztam meg, melynek tükrében előfeldolgozó eljárásokat is fejlesztettem. Munkám során azt is figyelembe vettem, hogy egyes üzleti folyamatok (pl. logisztikai tevékenység) nem Petri-hálóként működnek, hanem elengedhetetlen a folyamatlépések időbélyegeinek figyelembevétele is. Az általunk kidolgozott fuzzy csoportosítási algoritmus a folyamatbányászati algoritmusok eredményeinek kiértékelését is segíti.

A dolgozatomban bemutatom az új folyamatbányászati, elő- és utó-feldolgozási algoritmusok fejlesztésével kapcsolatos eredményeim, a létrehozott MATLAB eszköztárat, és esettanulmányok segítségével igazolom a kidolgozott eszközrendszer használhatóságát.

Kulcsszavak: Folyamatbányászat, Algoritmus fejlesztés, Matlab, Fuzzy

Abstract

The development and optimization of production and business processes require detailed models. The production support systems and ERPs record the most important pieces of information in connection with executing process steps into event log. Using process mining algorithms it becomes possible to generate and validate process models automatically based on these logs. Due to the complexity and stochastic nature of processes and event logs the identification of models are not trivial.

Process mining algorithms found in the technical literature are available within limits, however, their development is inhibited due the source code of these software are not available.

To overcome this difficulty I have developed a widely accessible toolbox of process mining through the MATLAB implementation of *Alpha*- and *Heuristic* algorithms. I have defined development possibilities to improve computation complexity and scalability, and concerning this I have developed pre-processing algorithms. During the work I also took into consideration that certain business processes (e.g. logistics activity) do not function as a Petri-network, so it is also necessary to take into consideration the time stamps of process steps. The fuzzy clustering algorithm created by me helps to evaluate also the results of the process mining algorithms.

In the thesis I introduce the results concerning the development of new process mining, pre- and post-processing algorithms, the created MATLAB tool set, and I prove the applicability of the created toolbox with the help of case studies.

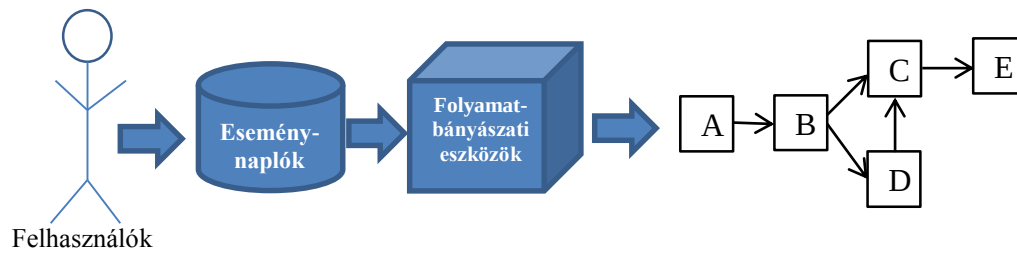
Keywords: Process mining, Algorithm development, Matlab, Fuzzy

1 Célkitűzés és irodalmi áttekintés

A versenyképesség folyamatos fenntartása és növelése érdekében fejlesztésekre van szükségünk. Ezekre a fejlesztésekre a folyamat pontos modelljének ismeretében szimulációval és optimalizációval tehetünk javaslatokat [1]. A folyamat-szimuláció alapú folyamatelemzésre a korábbi kutatásaink során már kidolgoztunk egy keretrendszert [2, 3]. Az akkori munkánk során feltártuk, hogy a folyamatmodellek feltérképezése korántsem egyszerű feladat. Jelen dolgozat célkitűzése a folyamatmodellek feltárására alkalmas, folyamatbányászati technikákon alapuló megoldások fejlesztése a rendszerek által generált eseménynaplók alapján.

Ma már minden nagyobb szervezet készít eseménynaplókat különböző folyamatairól. A gyártósorok, szállítási folyamatok és a különböző banki rendszerek működését támogató informatikai eszközök folyamatosan követik és eseménynaplókban rögzítik a folyamatváltozókat és a felhasználók tevékenységét. Ezen eseménynaplók alapján a rendszer működése ellenőrizhető, kiértékelhető, illetve a rejtett szervezeti struktúrák is feltárhatók [4]. Munkám első lépéseként ezeket a naplókat vizsgáltam, hogy az azokból kinyerhető információkból lehetséges-e folyamatmodelleket feltárni.

A folyamatmodellek eseménynapló alapú identifikálása azok hatalmas méretei és a folyamatok bonyolultsága miatt nem triviális feladat. Ennek a problémának a feloldására alakult ki az adatbányászat egy speciális ágaként a folyamatbányászat tudományterülete (1. ábra), elsősorban Wil M. P. van der Aalst [5] holland kutató és társai munkásságának köszönhetően. Az általuk fejlesztett ProM szoftvercsomag [6] számos folyamatbányászati algoritmust kínál, azonban a szoftvercsomagban megtalálható folyamatbányászati algoritmusok csak alkalmazhatók, magukhoz az algoritmusokhoz nem férhetünk hozzá, azokat közvetlenül fejleszteni nem tudjuk, így vált szükségessé ezeknek az algoritmusoknak a fejlesztése. Wil M. P. van der Aalst és társai tanulmányukban bemutatják az *Alpha* algoritmust [7], mint a folyamatbányászat legelső hatékony módszerét.



1. ábra. Folyamatbányászat struktúrája

A felhasználók, vagy a rendszerek által generált eseménynaplókból folyamatbányászati algoritmusok segítségével építjük fel a folyamatmodellt.

Az algoritmus során az eseménynaplókból feltárjuk az egyes folyamatlépéseket és az azok közötti kapcsolatot. A cikk bemutatja az eljárás definícióit és egy egyszerű példán be is mutatja működését. Részletezik az algoritmus hátrányait, melyek szerint az általa feltárt modellek egyes esetekben túlságosan is részletesek.

Az *Alpha* algoritmus hátrányát oldja fel a Holland csoport egy másik cikkükben, melyben a *Heurisztikus* algoritmus [8] használhatóságát bizonyítják. A cikk részletezi a folyamatok struktúrájának és felépítettségének az eseménynaplóktól való függését. A *Heurisztikus* algoritmus nagy előnye, hogy definiálhatunk egy határértéket, amellyel szabályozhatjuk a feltárt modell részletességét. A kutatócsoport cikkeiben kifejti, hogy az *Alpha* és *Heurisztikus* algoritmusok eredményeiből milyen módszerekkel tudjuk megalkotni a Petri-háló modelleket. A *Heurisztikus* bányászat esetén ezt egy másik cikkük ismerteti, hogy hogyan tudjuk megkülönböztetni az eseménynaplók alapján az ÉS, illetve a Kizáró VAGY kapcsolatokat [9].

Kutatásom során egy másik megközelítést is megvizsgáltam, Dolean és társai [10] az üzleti folyamatok bányászatára, az úgynevezett „Decision Process Data Models” (DPDM) módszert alkalmazták. Ez a módszer is egy speciális adatbányászati modell. A naplók alapján feltárja, és fa gráfként rögzíti az adatokat aszerint, hogy milyen szintű levélnek felelnek meg. Werner Ágnes és társai folyamatbányászati eljárásokat alkalmaztak modell alapú hiba kimutatásához. Eredményeiket [11] cikkben foglalják össze, ahol bizonyítják, hogy a folyamatbányászat segítségével ezen hibák feltárhatók és elemezhetők. Egy másik munkában Werner Ágnes és Dulai Tibor [12] folyamatbányászati megközelítéssel tárnak fel funkcionális hibákat járműipari folyamatok esetében.

Jiexun Li és társai tanulmányukban bemutatják a folyamat-feltérképezés egy lehetséges módszertanát a hagyományos információszerzés (interjúk, megbeszélések) és az analitikus jellegű információszerzésen túl a folyamatbányászat lehetőségeit is figyelembe véve (Policy-Based Process Mining (PBPM)) [4]. Az üzleti folyamatok bányászati lehetőségeiről szintén Wil M. P. van der Aalst értekezik, melyben összefoglalja az ide vonatkozó *ProM* és *MXML* ismereteket és egy konkrét esettanulmányt is bemutat egy holland szervezet esetében [4].

A folyamatbányászat témakörében található rengeteg irodalom is azt mutatja, hogy egy gyorsan fejlődő tudományterületről van szó, melyben még rengeteg fejlesztési potenciál van. Kutatásom során a folyamatmodellek eseménynaplók alapján való feltárására a legalkalmasabbnak a Holland kutatócsoport által kidolgozott algoritmusokat találtam, azonban ezek az algoritmusok nem érhetőek el, így legfontosabb célkitűzésünk az *Alpha* és a *Heurisztikus* algoritmusok fejlesztése volt MATLAB környezetben, hogy a későbbi munkánk során ezeket az eljárásokat integrálni tudjuk további folyamatmenedzsmenti alkalmazásokkal.

Az *Alpha* algoritmus valós méretű problémák esetén a nagyszámú élek és csomópontok miatt szerteágazó és gyakran átláthatatlan, összetett modellt eredményez. Annak érdekében, hogy az eseménynaplók alapján feltárt folyamatmodelleket elemezni tudjuk, ezeket egyszerűsíteni, átláthatóbbá kell tennünk. Ennek egyik eszköze a folyamatlépések klaszterezése. E tekintetben a holland kutatócsoport már készített publikációkat [13, 14], melyekben áttekintést adnak az algoritmusról és azok felhasználhatóságáról is. Bemutatják, hogy erre a célra fuzzy csoportosítást alkalmaznak és definiálják folyamatlépések esetén a halmaz-hovatartozási értékeket is.

A piacon fellelhető további szoftverek (pl.: Disco [15], Celonis Process Mining [16], BPMone [17]) céljaimnak nem voltak megfelelőek, így annak érdekében, hogy a folyamatbányászati algoritmusok széles körben használhatóak és fejleszthetőek legyenek, MATLAB programcsomagban fejlesztettem azokat.

A fejlesztett rendszerrel a folyamatbányászati eszközök integrálhatóvá válnak további algoritmusokkal, melyek lehetővé teszik a folyamatok teljes vizsgálatán túl azok optimalizálását is.

A munkám legnagyobb eredménye az alap folyamatbányászati algoritmusok fejlesztése, hiszen ezek az algoritmusok eddig nem voltak elérhetőek, pusztán alkalmazni lehetett őket a piacon megtalálható szoftverek segítségével. Az algoritmusokról a megjelent irodalmak alapján pusztán azok matematikai háttéréről kaptam információt, magukat az algoritmusokat fejlesztenem kellett egy arra megfelelő programnyelven, melyre a MATLAB szoftvercsomagot választottam. Bár az algoritmusok leírása egyszerű és hatékony megoldást sejtet, azok gyakorlati megvalósítása rengeteg problémát vetett fel, és a hatványhalmazok alkalmazása miatt igen lassúnak is bizonyultak. Ezért már ezen alap algoritmusok esetén is több helyen fejlesztést, gyorsítást eszközöltem, melynek eredményeképpen a módszerek valós problémák megoldására is alkalmassá váltak, amellett hogy implementációkat ingyenesen elérhetővé téve, egy könnyen értelmezhető, jól működő megoldást kínálok a folyamatbányászati szakemberek számára, támogatva ezzel a módszerek további fejlesztését, és széles körben elérhetővé tételét. További nagyon fontos eredményeim közé tartozik ezeknek az algoritmusoknak a továbbfejlesztése, melyekről részletesen a 4. és 5. fejezetben fogok írni.

Dolgozatomban bemutatom a folyamatbányászat alap definícióit, felhasználási lehetőségeit. A harmadik fejezetben részletezem a folyamatbányászati algoritmusokat és mindegyik alkalmazására mutatok egy-egy rövid példát. A 4. fejezetben a folyamatmodellek vizualizációját mutatom be az általam fejlesztett módszerekkel. Az 5. fejezetben részletesen bemutatom az alap algoritmusok továbbfejlesztését és egy általam kidolgozott súlyozási mértékekkel bővített eljárást is. Dolgozatom végén rámutatok az időbélyegek kezelésének fontosságára, felvázolom a továbbfejlesztési elképzeléseimet, majd egy esettanulmányon keresztül bemutatom a kifejlesztett módszerek használhatóságát.

2 Algoritmus fejlesztés specifikációi

A folyamatbányászati algoritmusok fejlesztésének célja, hogy olyan szoftverrendszert hozzak létre, amelynek segítségével a folyamatnaplók alapján feltárhatom az adott rendszer folyamatmodelljét. A szoftver funkcióinak az alábbi pontokat kell tartalmaznia:

- *Alpha* algoritmus
- *Heurisztikus* algoritmus
- *Fuzzy* algoritmus
- Szabványos *xgml* fájlok generálása
- Integrálhatóság folyamat-optimaló rendszerekkel
- További fejlesztésekkel való integrálhatóság

2.1 A keretrendszer szoftvereinek megválasztása

Az algoritmusokat MATLAB környezetben fejlesztettem, mivel fontos szempont volt, hogy a fejlesztett rendszer integrálható legyen a korábban fejlesztett eszközrendszerrel [2,3].

A MATLAB a mérnöki szférában széleskörűen elterjedt rendszer, amely több programrendszert is magában foglal és a különböző eszközrendszerek egymással integrálhatóak. Ezeket az eszközrendszereket sok esetben különböző kutatócsoportok fejlesztik, így a munkám egyik célja az volt, hogy egy hivatalos folyamatbányászati eszközt fejlesszek. Belátható, hogy a MATLAB programcsomag választásával nem csak a korábbi rendszerekkel való integrálhatóság valósul meg, hanem későbbi fejlesztések is elérhetővé válnak, hiszen ezek a MATLAB kódok nyílt forráskódúak, így bárki által felhasználható és továbbfejleszthető.

A folyamatbányászati algoritmusokkal feltárt modellek leírásából elemezhető modelleket kell megrajzolnom, ehhez egy hozzáférhető, felhasználóbarát szoftvert kellett találnom. A választásom az *yWorks* által kifejlesztett *yEd* [21] szoftver, amely egy széleskörű gráf megjelenítő.

Az *yEd*-vel az alábbi funkciók érhetőek el:

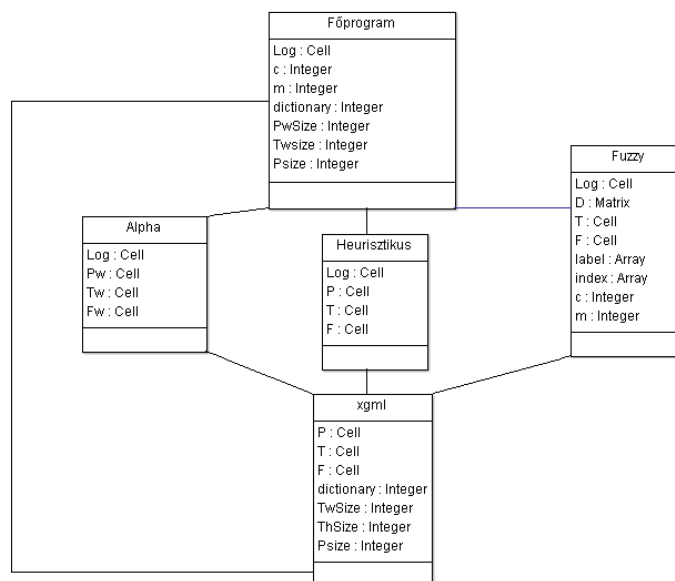
- Széleskörű folyamatmodellek (UML, BPMN)
- Automatikus strukturálás (amelynek segítségével a modellek elemei tetszés szerinti elrendezésbe felrajzolhatók)
- Szabványos *xgml* fájlok olvasása, exportálása
- Elemek csoportokba foglalása
- Modellek exportálása különböző formátumokban (bmp, pdf, png)

2.2 Szoftverrendszer felépítése

A szoftverrendszer 4 fő elemből áll össze, mely rendszert úgy kell kialakítani, hogy az a továbbiakban további algoritmusokkal bővíthető legyen:

- A rendszert indító főprogram
- *Alpha* algoritmust megvalósító program
- *Heurisztikus* algoritmust megvalósító program
- *Fuzzy* algoritmust megvalósító program
- Szabványos *xgml* fájlokat generáló program

A tervezett szoftverrendszer modelljét a 2. ábra mutatja, melyről leolvasható, hogy a különböző folyamatbányászati algoritmusokat a főprogrammal hívhatjuk meg és az algoritmusok eredményét átadva pedig az *xgml* fájlokat generáló programmal rajzolom meg.



2. ábra. A szoftverrendszer modellje

3 A folyamatbányászat elméleti eszköztára

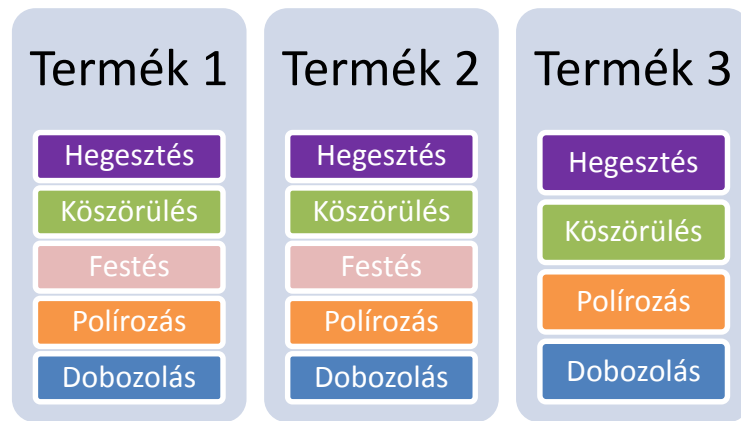
A folyamatbányászat [18] egy olyan technika, amellyel eseménynaplókból strukturált folyamatleírást készíthetünk; abból a célból, hogy feltárjuk, monitorozzuk és javítsuk a már meglévő valós folyamatainkat. A módszernek három fő alkalmazási lehetősége van:

- *Folyamatfeltárás*: a rendelkezésre álló adatokból feltárjuk a valós folyamatokat egy szervezet életében.
- *Eltérésvizsgálat*: a már korábban feltárt folyamatok modelljeihez viszonyíthatjuk az aktuális történéseket, ezáltal rendellenéseket azonosíthatunk.
- *Teljesítményelemzés*: különböző folyamatmodelleket hasonlíthatunk össze a tekintetben, hogy bizonyos változtatások milyen teljesítményváltozást eredményezhetnek.

A folyamatmodell [18] egy vizuális reprezentáció, amely megjeleníti az üzleti folyamatban fontos feladatokat, információt biztosít azok sorrendjéről, időzítéséről és az erőforrások hozzárendeléséről. Folyamatnak tekintjük az olyan cselekvések, vagy lépések egymás utáni sorozatát, melyek eredményeképpen eljutunk egy bizonyos végpontra [20] (3. ábra).

A folyamatbányászattal feltárt modellek különböző folyamat-elemzések alapjául szolgálnak, melyekkel széleskörűen elemezni tudjuk ezeket a rendszereket. A vállalatok körében a legelterjedtebb területe az üzleti folyamatmenedzsment (Business Process Management-BPM [19]) amely egy olyan határterülete az IT és menedzsment technológiáknak, amely magában foglalja mindazon eszközöket, módszereket, amelyekkel az operatív üzleti folyamatokat tervezzük, végrehajtjuk és szabályozzuk, beleértve az alkalmazottakat, szervezeteket, alkalmazásokat, háttér-dokumentációkat és egyéb információforrásokat is. Hasonló az alábbi területekhez:

- BAM (Business Activity Monitoring)
- BOM (Business Operations Monitoring)
- BPI (Business Process Intelligence)
- Adatbányászat



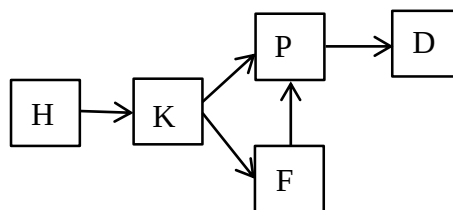
3. ábra. Termékek előállításának folyamatai

Termékek előállításának, fontosabb eseményeinek idősorrendi felírása.

A második ábra használatával vezetem be a folyamatbányászat alapgondolatait. A termékek előállítását leíró táblázatról első ránézésre elmondható, hogy a kezdő és a lezáró cselekvések, vagyis folyamatlépések mindig azonosak. Azonban jobban megvizsgálva látható, hogy az előállítás közbeni lépések sorrendje nem minden esetben azonos. Megfigyelhető, hogy az első két termék előállítási folyamatai azonos rutinon alapulnak, míg a harmadik termék folyamataiból kimarad a „Festés” folyamatlépés. Ez egy eseménynaplóként is felfogható, hiszen dokumentálja az adott eseteket (termékek), az adott feladatokat és ezek időbeni sorrendjét is. Ezek alapján az alábbi két szekvencia írható fel:

1. Hegesztés, Köszörülés, Festés, Polírozás, Dobozolás
2. Hegesztés, Köszörülés, Polírozás, Dobozolás

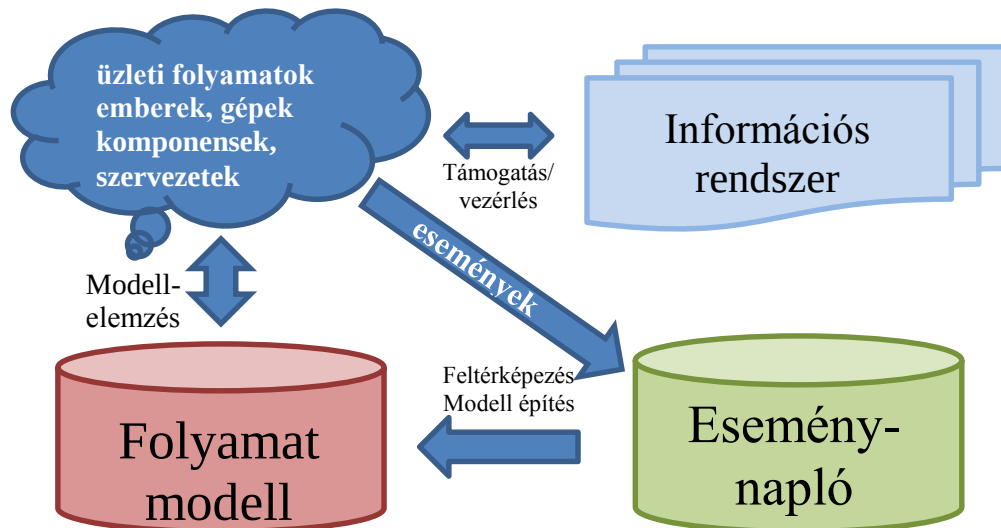
Ebben az egyszerű esetben már felrajzolható a folyamat modellje:



4. ábra. Termékek előállítási modellje

A harmadik ábrán definiált termék előállítási rutinfolyamatok modellje.

Amennyiben az előzőekben felírt folyamat egy összetettebb rendszer lenne, akkor a folyamatmodelljének felírása már nem lenne ennyire triviális feladat. Annak érdekében, hogy ezt a műveletet automatizálhassuk, folyamatbányászati módszereket alkalmazhatunk.



5. ábra. A folyamatmodellek és a „világ” kapcsolata

Ma már minden nagyobb rendszer rögzíti folyamatainak tevékenységeit és ezeket az eseménynaplókat használhatjuk fel a folyamatbányászatban.

A folyamatbányászat (5. ábra) célja a folyamatok adataiból valódi, hasznosítható tudásanyag kinyerése. A folyamatbányászat az adatbányászat egy speciális területe továbbá egy olyan folyamatmenedzsment technika, amely lehetővé teszi az üzleti folyamatok eseménynaplók alapján való elemzését. Segítségével felderíthetők és megjeleníthetők azok az összefüggések, amelyek a hosszú átfutási időért, a magas költségekért, vagy a rossz minőségért felelősek. Továbbá feltérképezhetők a gyártási folyamatban megjelenő hibák, amelyek a leállásokért, a hosszabb gyártási időért felelnek. Az alábbi kérdésekkel foglalkozik a folyamatbányászat:

1. Folyamat felfedezése, feltérképezése (Process discovery): mi történik valójában?
2. Hasonlóságelemzés (Conformance checking): az történik, amit elvárunk, amit előre meghatároztunk?
3. Teljesítményelemzés (Performance analysis): hol lehet javítani, változtatni?
4. Folyamat előrejelzés (Process prediction): vajon egy eset be fog következni?
5. Folyamatfejlesztés, tökéletesítés (Process improvement): hogyan alakítsuk át a folyamatot?

A folyamatbányászat alkalmazásának lehetőségei:

- Bankoknál
- Önkormányzatoknál
- Kormányzati ügynökségeknél
- Biztosításokhoz kapcsolódó társaságoknál
- Kórházakban
- Multi cégeknél
- Média vállalatoknál
- High-tech rendszerek gyártóinál és fogyasztóiknál

A folyamatbányászat általunk is tárgyalt algoritmusai:

- Alpha algoritmus
- Heurisztikus bányászat

A folyamatbányászati eljárásoknak az eseménynaplókkal szemben támasztott követelményei közül három pontot emelhetünk ki:

- *Eset azonosítója*: különböztessük meg az egyes esetekhez tartozó lépéseket (pl.: termékkód, vagy tranzakciós szám).
- *Feladat azonosítója*: szükséges, hogy az egyes folyamatlépéseknek pontosan definiált azonosítója legyen (pl.: pénzfelvétel, minőségellenőrzés).
- *A feladatok elvégzésének sorrendje*: az elvégzett feladatok időbeni sorrendisége definiált.

Az eseménynaplók formai követelménye, hogy az esetek egymástól elkülönüljenek és valamilyen struktúra alapján épüljenek fel (1. táblázat).

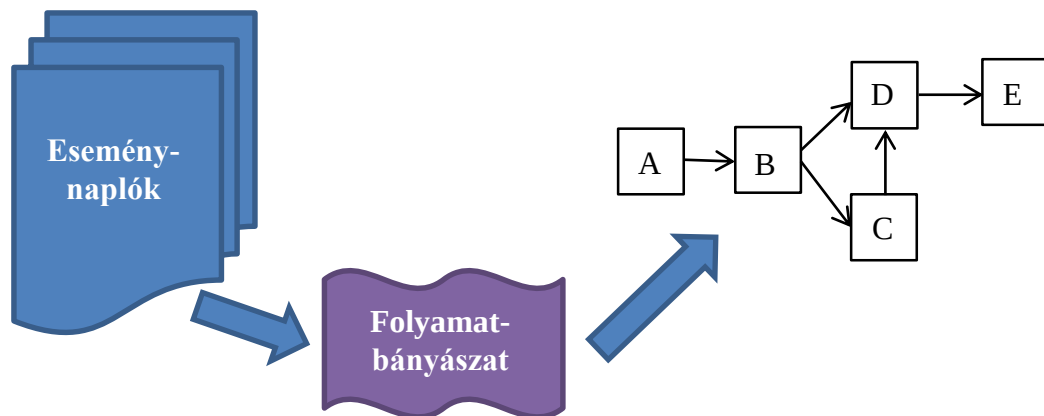
A folyamatbányászati algoritmusok segítségével bármely rendszerről modellezési információkat tudunk kinyerni (6. ábra). Az egyik első folyamatbányászati algoritmus az *Alpha* algoritmus [7], mely meghatározott matematikai definíciók szerint térképezi fel a folyamatok ok-okozati összefüggéseit a log fájlok alapján. Az *Alpha* algoritmus segítségével feltárható az adott folyamat minden lépése és a feladatok közötti sorrendiségi viszony is. Az egyik legfőbb hátránya, hogy a modellek részletessége túlságosan összetett, nem képes figyelembe venni az egyes kapcsolatok fontosságát.

1. táblázat. Példa egy eseménynaplóra

Eset ID	Esemény ID	Dátum	Tevékenység	Folyamatidő	Felelős	Költség
1	35654423	2010. 12. 30. 11:02	Regisztrálás	77	Péter	50
	35654423	2010. 12. 31. 10:06	Átvizsgálás	97	Elizabet	400
	35654423	2011. 1. 5. 15:12	Jegy ellenőrzés	43	Tamás	100
	35654423	2011. 1. 6. 11:18	Döntés	14	Dóra	200
	35654423	2011. 1. 7. 14:24	Kérés elutasítása	68	Péter	200
2	35654483	2010. 12. 30. 11:32	Regisztrálás	90	Tamás	50
	35654483	2010. 12. 30. 12:12	Jegy ellenőrzés	30	Tamás	100
	35654483	2010. 12. 30. 14:16	Véletlenszerű vizsgálat	22	György	400
	35654483	2011. 1. 5. 11:22	Döntés	18	Dóra	200
	35654483	2011. 1. 8. 12:05	Kártérítés	93	Miklós	200

A *Heurisztikus* folyamatbányászati eljárás [8] segítségével átláthatóbb modellek tárhatók fel, ugyanis az algoritmus már figyelembe veszi az egyes élek gyakoriságát és azok súlyozását is. Ezen adatoknak köszönhetően már definiálhatunk bizonyos határértékeket, melyek segítségével a kevésbé gyakori kapcsolatok elhagyhatók, ezáltal a „mindennapok valóságának” megfelelő modellt alkothatunk. Tehát a nagyon ritka esetek nem jelennek meg a modellekben, ezáltal átláthatóbbak, könnyebben vizsgálhatóak lesznek.

A következő alfejezetekben bemutatom a folyamatbányászat e két algoritmusát.



6. ábra. Folyamatbányászat alkalmazása

A folyamatbányászati algoritmusok segítségével tudunk bármely rendszerről modellezési információkat kinyerni.

3.1 Alpha algoritmus

Az *Alpha* algoritmus [7] a folyamatbányászat egyik első, alap algoritmus, melynek célja rekonstruálni az oksági viszonyokat az események sorozatában. Az algoritmus a különböző eseménynaplók meghatározott értékei alapján tárja fel a rendszer modelljét, melyhez az alábbi három információra van szüksége:

- Eset azonosítója
- Feladat azonosítója
- A feladatok elvégzésének időbeni sorrendiségére

Az algoritmus négy különböző kapcsolatot tár fel a különböző feladatok között:

- Direkt sorrend:
 - $x <_w y$, ha valamely x esetet közvetlenül az y eset követi
- Okozati viszony:
 - $x \rightarrow_w y$, ha $y <_w x$ és nem $y <_w x$
- Párhuzamosság (ÉS kapcsolatok):
 - $x || y$, ha $x <_w y$ és $y <_w x$
- Választás (Kizáró VAGY kapcsolatok):
 - $x \# y$, ha nem $x <_w y$ és nem $y <_w x$

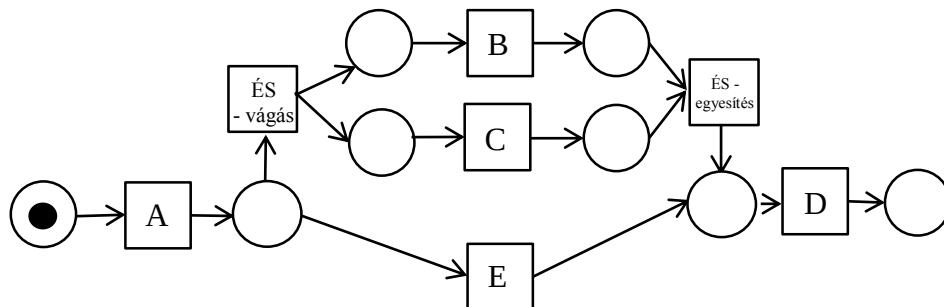
Az algoritmus részletes leírása:

A második táblázatban látható egy eseménynapló, ez az algoritmus kiindulási pontja. Az eseménynapló egy folyamatot ír le, melyet a 7. ábra mutat.

A 7. ábra által bemutatott folyamatmodell egy speciális Petri-háló, úgynevezett *WF-net*. Az algoritmus célja, az eseménynapló alapján feltárni az egyes feladatok közötti kapcsolatot és felépíteni a modellt. Az ún. helyeket a körök, míg az átmeneteket (feladatokat) a négyzetek jelképezik.

2. táblázat. A feldolgozandó eseménynapló

Eset azonosító	Feladat azonosító
1	A
2	A
3	A
3	B
1	B
1	C
2	C
4	A
2	B
2	D
5	A
4	C
1	D
3	C
3	D
4	B
5	E
5	D
4	D



7. ábra. Példa folyamatmodellre

A második táblázatban definiált eseménynapló folyamatmodellje.

Legyen W egy munkafolyamat a T eseménynaplóban. Az $Alpha(W)$ a következőképpen definiálható:

Első lépésként az algoritmus megvizsgálva a munkanaplót, feltölti a T_W feladatok halmazát. Majd a második és a harmadik lépésben meghatározza a teljes rendszernek a be- és kimeneteit (T_I és T_O):

$$T_W = \{t \in T \mid \exists_{\sigma \in W} t \in \sigma\} \quad (1)$$

Alpha algoritmus

Legyen $\sigma = a_1, a_2, \dots, a_n \in A^*$ sorozat, legyen A az eseteket azonosító ABC halmaza és n az A halmaz elemszáma. A *first* és *last* operátorokat az alábbiakban definiáljuk:

1. $a \in \sigma$, ha $a \in \{a_1, a_2, \dots, a_n\}$,
2. $first(\sigma) = a_1$, ha $n \geq 1$, és
3. $last(\sigma) = a_n$, ha $n \geq 1$

Tehát a *first* és a *last* műveletek meghatározzák egy folyamat legelső, és legutolsó elemét. Bemenetek:

$$T_I = \{t \in T \mid \exists_{\sigma \in W} t = first(\sigma)\} \quad (2)$$

Kimenetek:

$$T_O = \{t \in T \mid \exists_{\sigma \in W} t = last(\sigma)\} \quad (3)$$

A (4) és (5) lépésben az algoritmus meghatározza az X_W és Y_W halmazokat. Az X_W az egyes átmenetek okozati összefüggéseit határozza meg, tehát minden egyes (A, B) az X_W -ben úgynevezett átmenet-halmaz párokat hoz létre, ahol az első halmazból minden elem elérhető közvetlenül a másodikban, de a halmazokon belül nincs közvetlen kapcsolat az elemek között. Okozati kapcsolat van A -ból B -be, de A és B elemei sosem jelennek meg közvetlenül egymás után.

$$X_W = \left\{ (A, B) \mid A \subseteq T_W \wedge A \neq \emptyset \wedge B \subseteq T_W \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_W b \wedge \forall_{a_1, a_2 \in A} a_1 \#_W a_2 \wedge \forall_{b_1, b_2 \in B} b_1 \#_W b_2 \right\} \quad (4)$$

Az Y_W -ben az algoritmus finomítja az X_W beállításait azáltal, hogy ezek zárt halmazok (átmenet-párok) lesznek, hiszen nincsen az X_W -ben valódi részhalmazuk.

$$Y_W = \left\{ (A, B) \in X_W \mid \forall_{(A', B') \in X_W} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B') \right\} \quad (5)$$

A (6) lépésben jönnek létre a helyek, ahol az Y_W -t bővítjük ki a kezdő és végállapoti helyekkel.

$$P_W = \{P_{(A,B)} \mid (A, B) \in Y_W\} \cup \{i_w, o_w\} \quad (6)$$

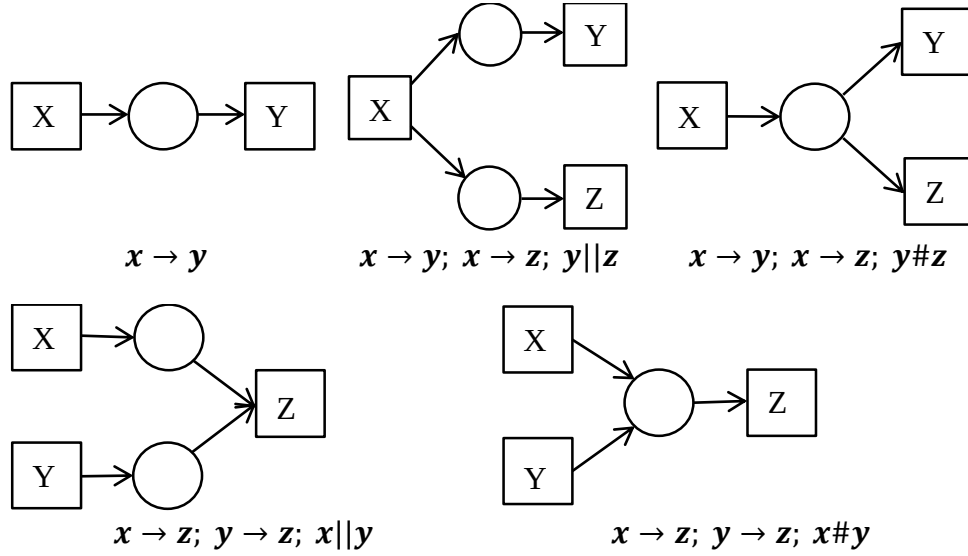
A helyeket és átmeneteket összekötő élek meghatározása történik az alábbi lépésben. Egy él egy átmenetet és egy helyet tartalmazó pár, ami az első elemből a másodikba mutat, tehát irányított éleket definiálunk.

$$F_W = \left[\begin{aligned} &\{(a, p_{(A,B)}) \mid (A, B) \in Y_W \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A, B) \in Y_W \wedge b \in B\} \\ &\cup \{(i_w, t) \mid t \in T_I\} \cup \{(t, o_w) \mid t \in T_O\} \end{aligned} \right] \quad (7)$$

Végül a (8) lépésben megadjuk a feltérképezett munkafolyamatot, melyet a helyek, a tranzakciók és az élek hármásával definiálunk.

$$Alpha(W) = (P_W, T_W, F_W) \quad (8)$$

A kapcsolatok ábrázolási lehetőségei az alábbiak:



8. ábra. A kapcsolatok megjelenítése

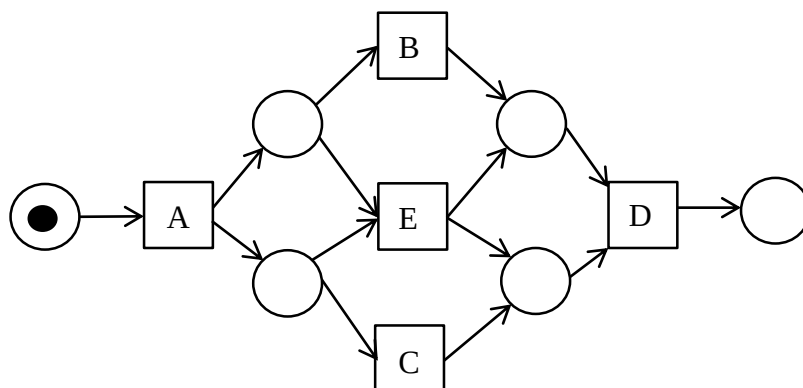
A különböző kapcsolatoknak, kapcsolati pároknak a lehetséges reprezentálása

Példa az algoritmus alkalmazására

Az algoritmust az 2. táblázatban látható eseménynaplón keresztül mutatjuk be, ahol $W = \{ABCD, ACBD, AED\}$. Az eseménynapló 3 esetet (1, 2, 3) és 5 különböző feladatot (A, B, C, D, E) tartalmaz. Az algoritmus 8 lépése az alábbi:

1. $T_W = \{A, B, C, D, E\}$
2. $T_I = \{A\}$
3. $T_O = \{D\}$
4. $X_W = \left\{ \begin{array}{l} (\{A\}, \{B\}), (\{A\}, \{C\}), (\{A\}, \{E\}), \\ (\{B\}, \{D\}), (\{C\}, \{D\}), (\{E\}, \{D\}), \\ (\{A\}, \{B, E\}), (\{A\}, \{C, E\}), (\{B, E\}, \{D\}), \\ (\{C, E\}, \{D\}), \end{array} \right\}$
5. $Y_W = \{(\{A\}, \{B, E\}), (\{A\}, \{C, E\}), (\{B, E\}, \{D\}), (\{C, E\}, \{D\})\}$
6. $P_W = \left\{ \begin{array}{l} i_W, o_W, p_{(\{A\}, \{B, E\})}, p_{(\{A\}, \{C, E\})}, \\ p_{(\{B, E\}, \{D\})}, p_{(\{C, E\}, \{D\})} \end{array} \right\}$
7. $F_W = \left\{ \begin{array}{l} (i_W, A), (A, p_{(\{A\}, \{B, E\})}), \\ (p_{(\{A\}, \{B, E\})}, B) \dots, (D, o_W) \end{array} \right\}$
8. $Alpha(W) = (P_W, T_W, F_W)$, ezt mutatja a 9. ábra

Az *Alpha* algoritmus nem minden esetben a legtömörebb modellt eredményezi. Nem képes figyelembe venni a különböző kapcsolatok gyakoriságát. Ennek tekintetében nem is képes kiszűrni a felesleges utakat, kapcsolatokat.



9. ábra. A feladat folyamatmodellje

A második táblázatban felírt eseménysorozatot megvalósító folyamatmodell.

A következő alfejezetben bemutatom, hogy a *Heurisztikus* algoritmussal letisztultabb modelleket alkothatunk.

3.2 Heurisztikus bányászat

A *Heurisztikus bányászat* célja rekonstruálni az oksági viszonyokon túl azok gyakoriságának mértékét is az események sorozatában. A *Heurisztikus bányászat* során is az *Alpha* algoritmushoz hasonló eseménynaplókból tárjuk fel a rendszer modelljét, azonban van egy fontos különbség azok kiértékelésében, hiszen míg az *Alpha* algoritmusnál nem vettük figyelembe két *Feladat* közötti átmenet gyakoriságát, addig ebben az esetben ezt is rögzíteni és kezelni kell. Az így előálló gyakorisággal súlyozott adjanencia mátrixok segítségével már meghatározhatunk bizonyos határértékeket annak érdekében, hogy a keletkező modellünk kellőképpen tömör és valós legyen.

Összesen két határértéket definiálhatunk az algoritmus paraméterezése során:

- Meghatározhatjuk az élek előfordulási gyakoriságának a határértékét. Tehát definiálhatunk egy határt, amely alatti gyakoriság esetén az adott él megszűnik.
- A másik határérték az élek súlyozottságának definiál egy küszöböt, mely alatt az adott él szintén eltűnik.

Ezeknek a határértékeknek a definiálásával lehetőségünk nyílik az *Alpha* algoritmus hiányosságainak kiküszöbölésére és ezáltal pontosabb modelleket alkotni.

A heurisztikus folyamatbányászat esetén az eddigi ismereteinket (az *Alpha* algoritmusnál definiáltakat) néhány fontos új definícióval egészítjük ki:

$W: |A >_W B|$ megadja, hogy az A hányszor követi összesen közvetlenül az B -t a W eseménynaplóban.

$$|A >_W B| = \sum_{\sigma \in W} W(\sigma) \times |\{1 \leq i < |\sigma| \mid \sigma(i) = A \wedge \sigma(i+1) = B\}| \quad (9)$$

A kapcsolati mátrix alapján az alábbi definícióval határozzuk meg az egyes lépések közötti kapcsolatok típusát:

$$A \Rightarrow_W B \wedge C = \left(\frac{|B >_W C| + |C >_W B|}{|A >_W B| + |A >_W C| + 1} \right) \quad (10)$$

Amennyiben ez az érték egy meghatározott határérték (szakirodalomban 0.1 [9]) alatt van, akkor a két folyamatlépés között Kizáró VAGY (XOR) kapcsolatot, abban az esetben pedig, ha nagyobb, akkor ÉS kapcsolatot definiálunk.

$|A \Rightarrow_W B|$ megadja a függőségi viszonyát A -nak és B -nek az alábbi módon:

$$|A \Rightarrow_W B| = \begin{cases} \left(\frac{|A >_W B| - |B >_W A|}{|A >_W B| + |B >_W A| + 1} \right), & \text{ha } A \neq B \\ \frac{|A >_W A|}{|A >_W A| + 1}, & \text{ha } A = B \end{cases} \quad (11)$$

Az algoritmust a (12). egyenletben felírt eseménysorozat alapján mutatom be. Ennek az eseménysorozatnak a 3. táblázatban felírt eseménynapló felel meg. Az eseménynapló 40 *Esetet* (1, 2, 3,...40) és 5 különböző *Feladatot* tartalmaz.

$$L = \left[(A, E)^5, (A, B, C, E)^{10}, (A, C, B, E)^{10}, (A, B, E)^1, (A, C, E)^1, \right. \\ \left. (A, D, E)^{10}, (A, D, D, E)^2, (A, D, D, D, E)^1 \right] \quad (12)$$

3. táblázat. A (12). egyenletben felírt eseménynapló

1 – A	10 – C	17 – B	24 – B	33 – D
1 – E	10 – E	17 – E	24 – E	33 – E
2 – A	11 – A	18 – A	25 – A	34 – A
2 – E	11 – B	18 – C	25 – C	34 – D
3 – A	11 – C	18 – B	25 – B	34 – E
3 – E	11 – E	18 – E	25 – E	35 – A
4 – A	12 – A	19 – A	26 – A	35 – D
4 – E	12 – B	19 – C	26 – B	35 – E
5 – A	12 – C	19 – B	26 – E	36 – A
5 – E	12 – E	19 – E	27 – A	36 – D
6 – A	13 – A	20 – A	27 – C	36 – E
6 – B	13 – B	20 – C	27 – E	37 – A
6 – C	13 – C	20 – B	28 – A	37 – D
6 – E	13 – E	20 – E	28 – D	37 – E
7 – A	14 – A	20 – A	28 – E	38 – A
7 – B	14 – B	20 – C	29 – A	38 – D
7 – C	14 – C	20 – B	29 – D	38 – D
7 – E	14 – E	20 – E	29 – E	38 – E
8 – A	15 – A	21 – A	30 – A	39 – A
8 – B	15 – B	21 – C	30 – D	39 – D
8 – C	15 – C	21 – B	30 – E	39 – D
8 – E	15 – E	21 – E	31 – A	39 – E
9 – A	16 – A	22 – A	31 – D	40 – A
9 – B	16 – C	22 – C	31 – E	40 – D
9 – C	16 – B	22 – B	32 – A	40 – D
9 – E	16 – E	22 – E	32 – D	40 – D
10 – A	17 – A	23 – A	32 – E	40 – E
10 – B	17 – C	24 – C	33 – A	

A példa alapján az alábbi 8 különböző szekvenciát különböztethetjük meg:

1. AE
2. ABCE
3. ACBE
4. ABE
5. ACE
6. ADE
7. ADDE
8. ADDDE

Ahhoz, hogy felépíthessük a modellt, szükségünk van egy függőségi mátrixra (adjacencia mátrix). A heurisztikus eljárás során feltárjuk az egyes *Feladatok* közötti kapcsolatok gyakoriságának a számát is:

4. táblázat. A gyakorisággal súlyozott adjacencia mátrix

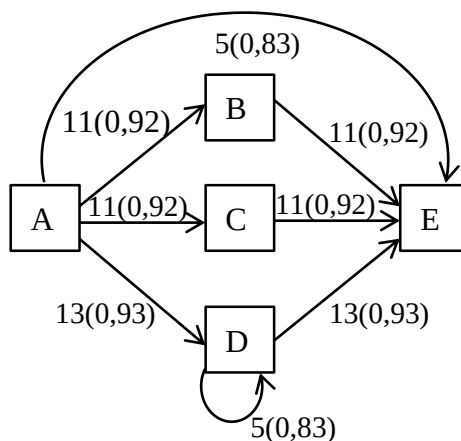
$ >_L $	A	B	C	D	E
A	0	11	11	13	5
B	0	0	10	0	11
C	0	10	0	0	11
D	0	0	0	4	13
E	0	0	0	0	0

Ahhoz, hogy az egyes kapcsolatok súlyozásának mértékét megállapíthassuk, az előzőekben már részletezett (11) összefüggést alkalmazzuk. Ez alapján felírható a súlyozott adjacencia mátrix:

5. táblázat. Súlyozott adjacencia mátrix

$ \Rightarrow_w $	A	B	C	D	E
A	$\frac{0}{0+1} = 0$	$\frac{11-0}{11+0+1} = 0,92$	$\frac{11-0}{11+0+1} = 0,92$	$\frac{13-0}{13+0+1} = 0,93$	$\frac{5-0}{5+0+1} = 0,83$
B	$\frac{0-11}{0+11+1} = -0,92$	$\frac{0}{0+1} = 0$	$\frac{10-10}{10+10+1} = 0$	$\frac{0-0}{0+0+1} = 0$	$\frac{11-0}{11+0+1} = 0,92$
C	$\frac{0-11}{0+11+1} = -0,92$	$\frac{10-10}{10+10+1} = 0$	$\frac{0}{0+1} = 0$	$\frac{0-0}{0+0+1} = 0$	$\frac{11-0}{11+0+1} = 0,92$
D	$\frac{0-13}{0+13+1} = -0,93$	$\frac{0-0}{0+0+1} = 0$	$\frac{0-0}{0+0+1} = 0$	$\frac{4}{4+1} = 0,80$	$\frac{13-0}{13+0+1} = 0,93$
E	$\frac{0-5}{0+5+1} = -0,83$	$\frac{0-11}{0+11+1} = -0,92$	$\frac{0-11}{0+11+1} = -0,92$	$\frac{0-13}{0+13+1} = -0,93$	$\frac{0}{0+1} = 0$

Az alábbi, 10. ábra látható függőségi gráf a 5. táblázat4. táblázatban felírt függőségeket tartalmazza:

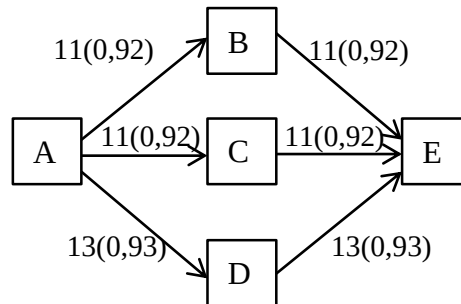


10. ábra. Súlyozott folyamatmodell

A teljes modell, az éleken feltüntetve a súlyozási értékek.

A 11. ábra bemutatja, hogy függőségi gráf felrajzolásánál már meghatároztuk a határértékeket és ez által egyes élek kiestek. Az alábbi határértékeket definiálunk:

- $5 > |>_w|$
- $0,9 > |\Rightarrow_w|$



11. ábra. Letisztított súlyozott modell

A 10. ábra által mutatott folyamatmodellnek, a határértékek alapján egyszerűsített változata.

3.3 Folyamatlépések csoportosítása

A dolgozatban a *Fuzzy Miner* [13] algoritmust vizsgáltuk, amely folyamatok gráf alapú klaszterezésére alkalmazható. A folyamatot szimbolizáló gráfban csúcsok reprezentálják a folyamat lépéseit (események, pl. rendelés, teljesítés), az azokat összekötő élek pedig az egyes események sorrendi relációit mutatják.

Az egyes folyamatok elemzése során azért döntöttünk a fuzzy algoritmus alkalmazása mellett, mert a folyamatbányászati algoritmusok gyakran csak a kellően strukturált (és tisztított) folyamatoknál adnak jól értelmezhető eredményt, eltérő esetben nagyon gyakran úgynevezett spaghetti modelleket kapunk, ahol a gond nem a kapott modell pontosságával, hanem áttekinthetetlenségével van. Az ilyen kimenetek esetében nincs lehetőség az eredmény absztrakciójára (pl. egy térkép is csak akkor hasznos számunkra, ha a nagyobb városok máshogy vannak jelölve, mint a falvak, illetve az utak esetében is jól elkülöníthető egy autópálya az alsóbb rendű utaktól). A *Fuzzy Miner* eszközzel több eseménylépésre tekintve is megállapíthatóak az események közti összefüggések, pl. $A \rightarrow B$ és $B \rightarrow C$ kapcsolatokból $A \rightarrow C$ kapcsolatot is lehet elemezni.

Ebben az alfejezetben részletezem az általunk fejlesztett *Fuzzy miner* algoritmust, továbbá kitérek a fuzzy klaszterezés alaptételeire is.

Fuzzy halmazelmélet

A fuzzy halmaz a klasszikus halmaz általánosítása, vagyis annak továbbgondolása hogy egy halmaznak egy objektum eleme-e, vagy nem. Ezt a relációt az alábbi karakterisztikus függvénnyel reprezentálhatjuk:

$$X_x(x) = \begin{cases} 1, & \text{ha } x \in X \\ 0, & \text{ha } x \notin X \end{cases} \quad (13)$$

A fuzzy elmélet alapja a karakterisztikus függvénnyel való leírási mód megváltoztatása, eszerint a halmazfogalom általánosítása. Ahhoz, hogy az ún. halmaz-hovatartozási, vagy más néven tagsági függvényt definiálhassuk, tekintsük az alábbi halmazt:

$$F = \{(x, \mu_F(x)) : x \in X\} \quad (14)$$

Ezt az X feletti fuzzy halmaznak nevezzük, ahol az x elemek egy 0 és 1 közötti számmal jellemezhetőek, vagyis egy elem minél közelebb van az 1-hez, annál inkább tartozik a halmazhoz. Ezt nevezzük tagsági függvénynek:

$$\mu_F(x) : X \rightarrow [0,1] \quad (15)$$

Ahhoz, hogy jobban megérthessük és átláthassuk a Fuzzy logikát, nézzük az alábbi példát, amit az irodalom *Zadeh példájaként* említ [16].

Tekintsük az alábbi három fogalmat, és rendeljük hozzájuk halmaz definíciót:

- fiatal – F (x)
- középkorú – K (x)
- idős – I (x)

Az életkorokat tekintve mindegyik halmazhoz hozzárendelhetjük a karakterisztikus függvényét:

Fiatal:

$$x_{F(x)} = \begin{cases} 1, & \text{ha } x \leq 30 \\ 0, & \text{különben} \end{cases} \quad (16)$$

Középkorú:

$$x_{K(x)} = \begin{cases} 1, & \text{ha } 30 < x \leq 50 \\ 0, & \text{különben} \end{cases} \quad (17)$$

Idős:

$$x_{I(x)} = \begin{cases} 1, & \text{ha } x > 50 \\ 0, & \text{különben} \end{cases} \quad (18)$$

A fuzzy elméletben a tagsági függvényt rendeljük a fogalmakhoz, szemben a klasszikus halmaz karakterisztikus függvényével, ahol 0-t, vagy 1-et rendel az x -hez attól függően, hogy x eleme-e a halmaznak.

Fuzzy C-medoid klaszterezés

A Fuzzy algoritmus a *Heurisztikus* algoritmus által definiált súlyozott függőségi mátrix adataiból kiindulva végzi el a klaszterezést. A tagsági függvényt az eseménynapló alapján definiálja, az élek előfordulási gyakoriságának függvényében (21. egyenlet). Ezek az adatok alapján tudja eldönteni a csomópontokról, hogy egy csoportba tartoznak-e, és ezáltal összevonhatók-e, vagy sem.

Az algoritmus működésének bemutatásához először definiálnunk kell az alábbi jelöléseket:

- E – események (events)
- I – tevékenység esetek (activity instances)
- A – tevékenységek (activities)
- C - esetek (cases)
- $t: E \rightarrow R_0^+$ - minden eseményhez egy időbélyeget rendel hozzá
- $i: E \rightarrow I$ – minden eseményhez egy tevékenység esetet rendel hozzá
- $a: I \rightarrow A$ – minden tevékenység esethez hozzárendel egy tevékenységet
- $c: I \rightarrow C$ – minden tevékenység esethez hozzárendel egy esetet

Eseményöröklés (Event Succession)

Jelöljük az eseménynaplónkat W -vel, ekkor:

$$W = (E, I, A, C, t, i, a, c) \quad (19)$$

Definiálunk egy olyan értéket, amely annak a számosságát adja meg, hogy két esemény hányszor követi egymást a naplóban:

$$>_w: A \times A \rightarrow \mathbb{N} \quad (20)$$

Legyen az $a_i, a_j \in A$, ekkor a két esemény öröklésének számosságát az alábbi módon írhatjuk fel:

$$\begin{aligned} >_w(a_i, a_j) &= \#_{e_i, e_j \in E} (t(e_i) < t(e_j) \wedge a(i(e_i))) = a_i \wedge c(i(e_j)) \\ &= a_j \wedge c(i(e_i)) = c(i(e_j)) \wedge \#_{e_k \in E} (c(i(e_k))) \\ &= c(i(e_i)) \wedge t(e_i) < t(e_k) < t(e_j) \end{aligned} \quad (21)$$

Az $a_i >_w a_j$ helyett használjuk a $>_w(a_i, a_j) > 0$ jelölést.

A hasonlóság definiálja, hogy milyen gyakran követik egymást az események közvetlenül a naplóban. Ha a vonatkozó események egymást gyakrabban követik, akkor a hasonlóság növekszik.

Megjegyzendő, hogy ha a két tevékenység a_1 és a_2 különbözik, akkor ez az érték soha nem egyezik:

$$>_w(a_1, a_2) \neq >_w(a_2, a_1) \quad (22)$$

Tevékenységhasonlóság (Activity Similarity)

Definiáljunk egy hasonlósági értéket két tevékenység között:

$$\sigma: A \times A \rightarrow (0,1) \quad (23)$$

Ha $a_1 = a_2$, akkor $\sigma(a_1, a_2) = 1$, egyébként:

$$\sigma(a_i, a_j) = \frac{>_W(a_i, a_j) + >_W(a_j, a_i) + 1}{1 + 2 \max_{a_k, a_l \in A} (>_W(a_k, a_l))} \quad (24)$$

Ebből a hasonlósági függvényből határozhatjuk meg a Fuzzy klaszterezéshez szükséges tagsági halmazt.

Tagsági halmaz valószínűség (Cluster membership Probability)

Legyen $A^k \subseteq A$, és $|A^k| = k$ halmaza a medoid-nak, minden $a_i \in A^k$ és $a_j \in A$. A Egy klaszter legjellemzőbb pontjait nevezzük medoid-nak. Definiáljunk egy valószínűségi változót: $u(a_i, a_j)$, annak a valószínűségére, hogy a_j abba a klaszterbe tartozik, melynek középpontja az a_i „medoid”, vagyis $u: A^k \times A \rightarrow [0,1]$, ahol

$$u(a_i, a_j) = \frac{\sigma(a_i, a_j)^{\frac{1}{m-1}}}{\sum_{a_k \in A^k} \sigma(a_k, a_j)^{\frac{1}{m-1}}} \quad (25)$$

Az $m \in [1, \infty)$ tagot a szakirodalom „fuzzifier” tényezőnek nevezi [12]. Ezt általában $m = 2$ értéken rögzítjük, ekkor:

$$u(a_i, a_j) = \frac{\sigma(a_i, a_j)}{\sum_{a_k \in A^k} \sigma(a_k, a_j)} \quad (26)$$

Fuzzy k-Medoid algoritmus

Legyen $0 < k \leq |A|$ a kívánt klaszterek száma. Keresünk azon medoidokat $A_k \subseteq A$, ahol $|A_k| = k$, oly módon, hogy ez a halmaz minimalizálja az alábbi összefüggést:

$$\sum_{a \in A} \sum_{a^k \in A^k} \frac{u(a^k, a)^m}{\sigma(a, a^k)} \quad (27)$$

Az algoritmus nem garantálja, hogy megtaláljuk a globális minimumot. Ezen túlmenően, az eredmény függ a kezdeti medoidok véletlenszerű kiválasztásától, tehát az algoritmus nem-determinisztikus.

A klaszterezési eljárásokat is Matlab környezetben implementáltuk, ezáltal elérhetővé vált azok folyamatbányászattal való integrálása. Több módszert is teszteltünk, a Matlab-

hoz elérhető toolboxokkal és általunk implementált klaszterezési algoritmusokkal is elemeztük a feladatokat.

FCMD klaszterezés

Az FCMD algoritmus a folyamatbányászatban használatos klaszterezési eljárás, melyet munkánk során mi is fejlesztettünk MATLAB környezetben, hiszen ez az eljárás sem volt elérhető számunkra. Az előzőekben ismertetett FCMD algoritmust a folyamatbányászattal feltárt okozati mátrixszal paraméterezzük, az alábbi módon:

$$[\text{label}, \text{index}] = \text{FCMD_PM}(D, k, m)$$

D – távolsági mátrix

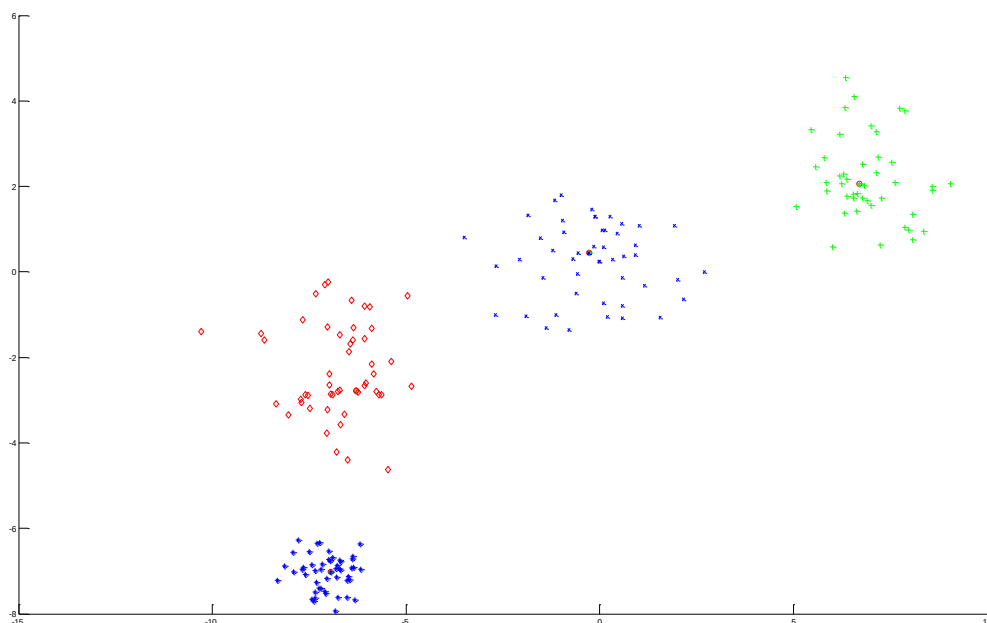
k – kívánt csoportok száma

m – Fuzzifier szám

A program eredményül az alábbi két értéket adja:

label – az egyes elemek hovatartozása

index – a kiválasztott csoportközpontok



12. ábra Az FCMD algoritmus

Az ábrán láthatjuk, hogy az algoritmus helyesen klaszterezi a pontokat.

Ezek alapján már látható, hogy mely pontok lettek csoportközpontok és mely pont melyik klaszterbe fog tartozni, tehát megkapjuk a hovatartozási halmazt. Egy egyszerű példán történő csoportosítást mutat a 12. ábra. Ebben az esetben a csoportok számát 4-nek választottuk, és az ábrázolásnál az egyes csoportokat külön-külön színnel jelöltük.

K-medoid klaszterezés

Az FCMD mellett a K-medoid algoritmust is implementáltuk, azonban a csoportosítás eredménye itt is függ a kezdeti állapotoktól. A programot az alábbi paraméterekkel hívhatjuk meg:

```
[label, index, cost] = kmedoids(D, k, init);
```

D – távolsági mátrix

k – kívánt csoportok száma

$init$ – opcionális érték, a klaszter középpontok kezdeti halmazát definiálhatjuk

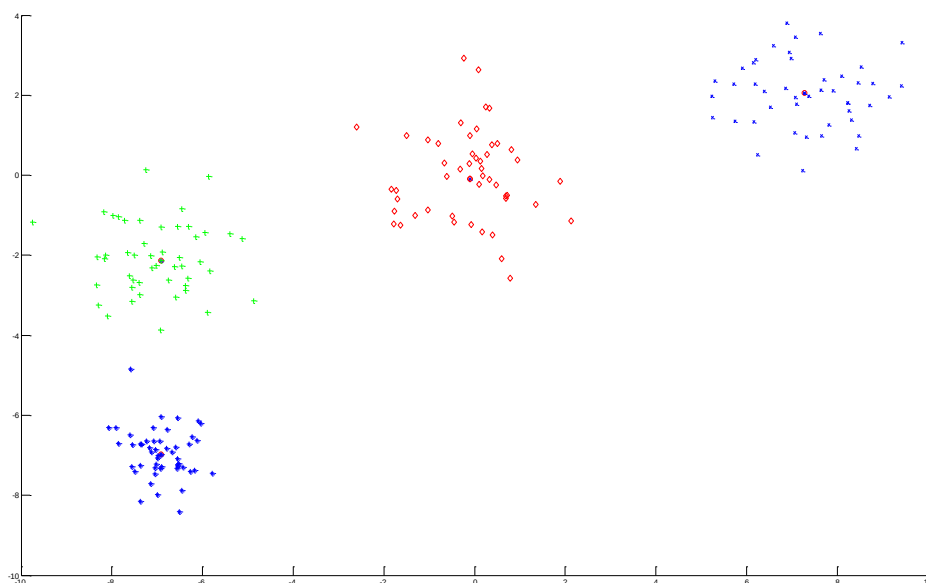
A program eredményül az alábbi három értéket adja:

$label$ – az egyes elemek hovatartozása

$index$ – a kiválasztott csoportközéppontok

$cost$ – a legjobb találat költségét, a távolságok összegét határozza meg

A 13. ábra mutatja a program eredményét, szintén 4 csoport felismerése esetén.



13. ábra. K-medoid algoritmus

Az eljárás klaszterezési eredményét láthatjuk, a középpontokat piros körrel jelöli.

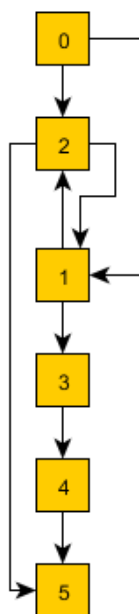
Munkám során fontos szempont volt, hogy a folyamatmodelleket reprezentáljuk. Erre a célra a következő fejezetben bemutatom az *yEd* [21] szoftvert.

4 Folyamatmodellek vizualizációja

A folyamatbányászati algoritmusok által feltárt modelleket *yEd* [21] szoftverrel vizualizálom. Ennek oka azon túl, hogy széleskörű folyamatmodell-típusok építésére

szolgál az, hogy képes ezeket a modelleket különböző módszerekkel strukturálni, továbbá „*xgml*” fájlformátummal dolgozik, amely nyílt forráskódú.

Az általam fejlesztett Matlab program a folyamatbányászati algoritmusok által feltárt adatokból generálja az „*xgml*” fájlokat az *yEd* számára. Az így kapott modellre alkalmazva a program automatikus strukturálását például az alábbi modellt kapjuk eredményül:

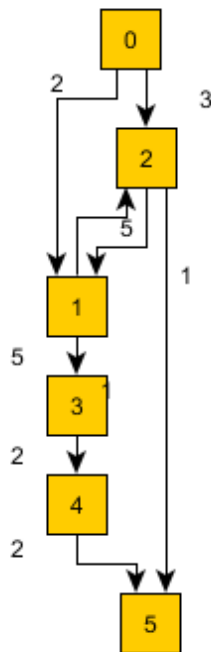


14. ábra. Egy kész folyamatmodell

Az *yEd* futtatása után előálló modell, hierarchikus elrendezésben.

4.1 Él gyakoriságok megjelenítése

A *Heurisztikus* folyamatbányászati algoritmuson túl, a következő fejezetekben bemutatom az *Alpha* algoritmuson végzett fejlesztéseim, ahol az élek gyakoriságának az ismerete és feltüntetése szintén nagyon fontos volt. Ezen szempontokat figyelembe véve úgy alakítottam át az „*xgml*” fájlokat generáló programot, hogy ezeket az értékeket meg tudjuk jeleníteni minden egyes él esetén, ezzel is elősegítve a modellek könnyebb átláthatóságát és vizsgálatát.

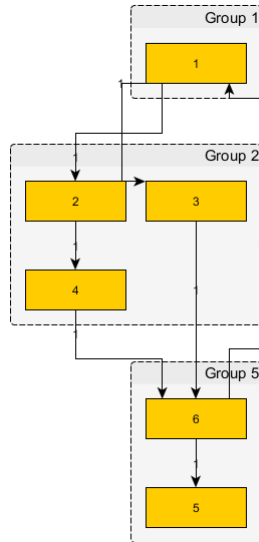


15. ábra

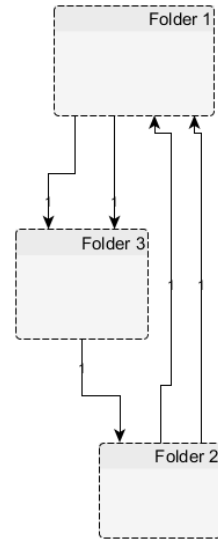
Az yEd modell az élek gyakoriságának megjelölésével

4.2 Folyamatok klaszterezésének megjelenítése

Ahhoz, hogy a folyamatok klaszterezését is vizualizálni tudjuk az *yEd* program segítségével, tovább kellett fejlesztenem az „*xgml*” fájlokat generáló programot és fel kellett tárnom a legjobb lehetőségeket ezeknek a klasztereknek a bemutatására. Az *yEd* program segítségével, az egyes folyamatlépéseket csoportokba tudjuk foglalni (16. ábra). Ezek után ezeket a csoportokat akár össze is tudjuk vonni, így a csoportokat ténylegesen is egy folyamatlépésként tüntetjük fel a modellben (17. ábra).



16. ábra. Csoportosított folyamatmodell
Látható, hogy az egyes folyamatlépéseket már csoportokba foglaltuk.



17. ábra. Az egyszerűsített folyamat-modell

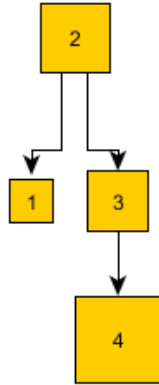
A 16. ábra által mutatott modellt már egyszerűsíteni tudtuk a tényleges csoportokra.

4.3 Súlyozások vizualizációja

Annak érdekében, hogy a súlyozási értékeket megfelelően megjelenítsem, úgy alakítottam át a programot, hogy abban az élek vastagsága és a folyamatlépések mérete változtatható legyen.

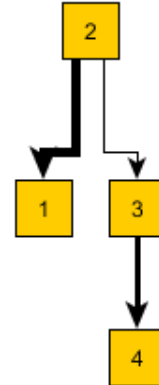
Hasonló módszerrel B.F. van Dongen és társa cikkében [13] találkozhatunk, de ők a módszert a hasonlósági hovatartozási mértékekre alkalmazták.

További súlyozásként tekinthetünk a *Heurisztikus bányászatban* definiált gyakoriságokra (9. egyenlet). További fejlesztés, hogy ezeket a gyakoriságokat, vagy él súlyokat is reprezentatívan jelenítem meg a modelleken, még hozzá az irányított élek (nyilak) vonalvastagságának a mértékével. A vonalvastagságot összesen 7 különböző mértékkel tüntetem fel (19. ábra).



18. ábra. Folyamatlépések súlyozása

Az ábrán látható, hogy a 4. folyamatlépés a „legsúlyosabb”, míg az első a legkevésbé.



19. ábra Élek súlyozása a folyamatmodellen

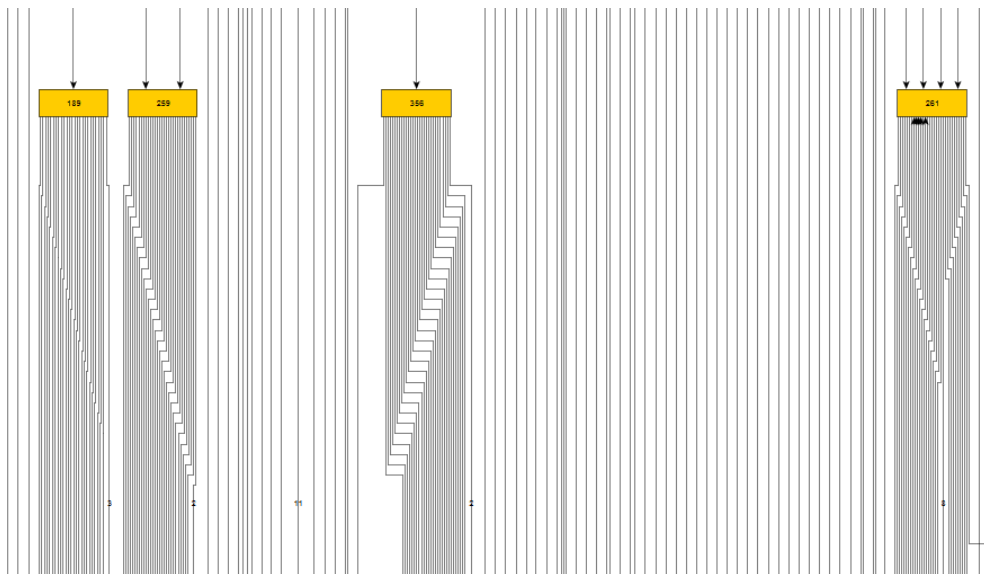
A modellen a „legsúlyosabb” él az egyes és kettes folyamatlépéseket összekötő él.

A következő fejezetben bemutatom, hogy milyen továbbfejlesztéseket dolgoztam ki az *Alpha* algoritmus esetén.

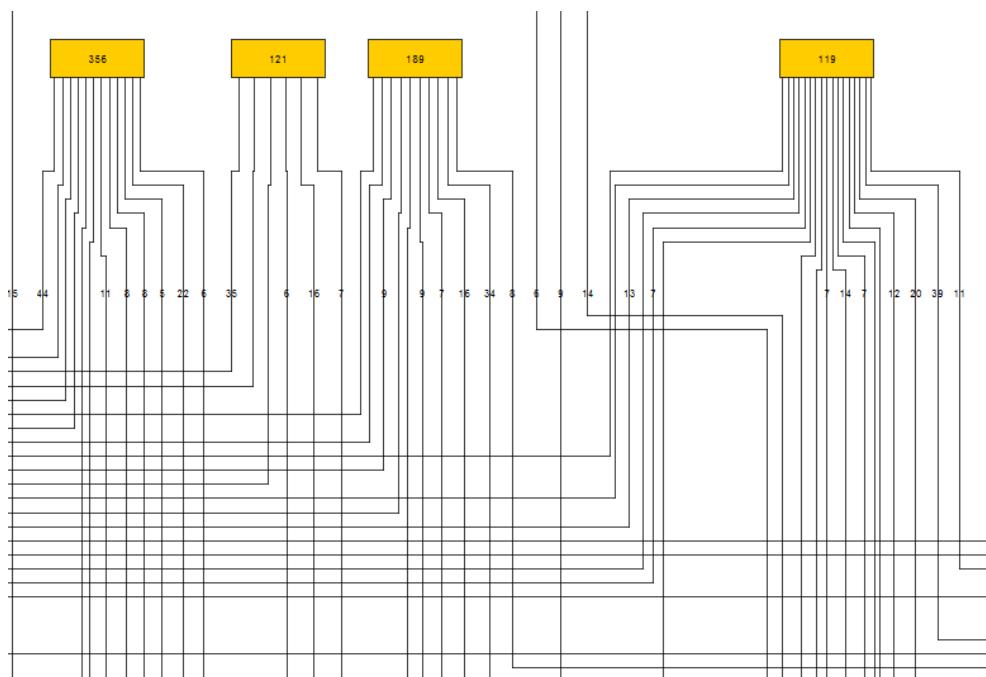
5 Alpha algoritmus fejlesztése

A harmadik fejezetben az *Alpha* algoritmus részletes bemutatásánál is rávilágítottam már arra, hogy a módszer gyakran terjedelmes és nagyon összetett, átláthatatlan modelleket eredményez, egy összetettebb rendszer esetén. Ennek elkerülése érdekében beláttam, hogy minden folyamatmodell esetén találhatóak olyan élek, amelyek a valóságban csak nagyon kivételes esetekben fordulnak elő, így a rendszer vizsgálata során ezek elhagyhatóak. Munkám során rájöttem, hogy hasonlóan a *Heurisztikus* folyamatbányászati módszerhez, az *Alpha* algoritmus esetén is célravezető lehet olyan határértékeket definiálni, mellyel a nem releváns élek, utak elhagyhatóak.

Az él-gyakoriságokat figyelembevevő fejlesztett *Alpha* algoritmusomat a már az irodalom alapján fejlesztett *Alpha* algoritmussal integráltam. Az algoritmus az *Alpha* eredményeit használja fel és a követési mátrix alapján a *Heurisztikus* bányászatnál definiált összefüggés (11) felhasználásával határozza meg minden egyes él gyakoriságát. A program paraméterezhető egy határértékkel, amely ezeknek a gyakoriságoknak a határértékét definiálja. Azokat az éleket, amelyek ezen érték alatt vannak, kivesszük az éleket definiáló F_w halmazból és a modellt ezután generáljuk le. A fejlesztett *Alpha* algoritmusra mutat példát a 20. ábra és a 21. ábra. Az ábrákon látható, hogy az első esetben egy átláthatatlan és rengeteg élt tartalmazó modelltől van szó, míg a második esetben egy letisztultabb és kevesebb éllel rendelkező modellt láthatunk. Természetesen a megfelelő modellt a határérték helyes paraméterezésével érhetjük el.



20. ábra. A teljes részletességű folyamatmodell részlete
Az ábrán látható, hogy rengeteg élt tartalmaz, amely egyes esetekben lehetetlenné teszi a modell vizsgálatát.



21. ábra. A fejlesztett Alpha algoritmus eredménye
Látható, hogy kevesebb élt tartalmaz a modell, így az elemzés elérhetőbbé vált. A megfelelő határérték megválasztásával fenntartható, hogy a modellben csak a valóságban ritkán előforduló élek ne jelenjenek meg.

További célkitűzésem volt, hogy a lefutását gyorsítsam. Ennek érdekében olyan eljárásokat javaslok, amelyekkel az eseménynaplók alapján olyan információkat tudunk feltárni, melyeket kezdőértékként átadva az algoritmusnak gyorsíthatjuk annak futási idejét. A legújabb kutatási területen olyan elő-feldolgozó klaszterezési eljárások fejlesztése, melyekkel az eseménynaplók alapján a folyamatlépéseket úgy csoportosíthatom, hogy melyek lehetnek párhuzamos események és ezekkel az információkkal lényegesen gyorsítani lehet az *Alpha* futási idejét.

6 Fejlesztett súlyozási módszerek

A heurisztikus és a klaszterezési algoritmusok áttekintése közben felmerült bennünk, hogy az élek elhagyása, illetve a csoportközéppontok kiválasztása esetén figyelembe kellene vennünk az egyes folyamatlépések további paramétereit. Ilyenek lehetnek pl.: folyamatidők, költségek, vagy akár az erőforrások hozzárendeltségei is.

Ebben az alfejezetben ezeknek a paramétereknek a figyelembevételét az általunk definiált súlyozási módszerekkel ismertetem, melyek új eredménynek számítanak és ezek segítségével pontosabb modelleket alkothatunk.

6.1 Súlyozás a heurisztikus folyamatbányászatban

Az előzőekben bemutatott *Heurisztikus bányászatnál* láthattuk, hogy ez a folyamatbányászati algoritmus a feltárt modellben elhagyja azokat az éleket, melyek egy bizonyos határt nem érnek el. Beláttam, hogy abban az esetben, ha egy él két fontos folyamatlépést köt össze, akkor az az él annak ellenére sem hagyható el, ha az a definiált értékeken kívül esik. Továbbá előfordulhat olyan eset is, hogy egy fontosnak ítélt folyamatlépés minden éle ezeken a határokon kívül esik, ekkor az a csomópont nem kerülne bele a modellbe. Ezekkel az esetekkel az *Heurisztikus bányászati* algoritmus nem foglalkozik, ezért ezeket a feltételeket definiálnom kellett.

Első lépésben meghatároztam a folyamatlépések *súlyát*, ezt az értéket az alábbi három paraméter alapján definiáltam:

- **Folyamatlépés ideje (PT)** – Az egyes folyamatlépések lefutási ideje különböző, minél hosszabb ez az idő, a folyamatlépésnek a kihasználtsága annál nagyobb lesz.
- **Erőforrásai (PR)** – A folyamatlépésekhez rendelt erőforrások fontos tényezők, egy erőforrás minél több folyamatlépéshez van rendelve, annál nagyobb lesz a kihasználtsága.
- **Költségei (PC)** – A folyamatlépések költségei nagymértékben hozzájárulnak azok súlyához.

A fenti paraméterek alapján az egyes folyamatlépések súlya (PMW):

$$PMW = PT + PR + PC \quad (28)$$

A folyamatlépés folyamatidejének súlyozási értéke, ahol $i = 1 \dots N$ és N a folyamatlépések száma, a *ProcessTime* pedig az adott folyamatlépés ideje:

$$PT_i = \frac{MAX(ProcessTime) + ProcessTime_i}{MAX(ProcessTime)} \quad (29)$$

A folyamatlépés erőforrás súlyozási értéke, ahol $i = 1 \dots N$ és N az egyes folyamatlépések száma, a *ProcessResource* pedig az adott folyamatlépés erőforrásának az összes folyamatlépéshez rendeltségének a száma (tehát pl.: ha egy munkagép három folyamatlépéshez van hozzárendelve, akkor *ProcessResource* = 3):

$$PR_i = \frac{MAX(ProcessResource) + ProcessResource_i}{MAX(ProcessResource)} \quad (30)$$

A folyamatlépés költségének súlyozási értéke, ahol $i = 1 \dots N$ és N az egyes folyamatlépések száma, a *ProcessCost* pedig az adott folyamatlépés költségének az értéke:

$$PC_i = \frac{\sum_{j=1}^N ProcessCost_j + ProcessCost_i}{\sum_{j=1}^N ProcessCost_j} \quad (31)$$

Az így definiált *PMW* érték alapján fejlesztettem tovább a *Heurisztikus bányászati* algoritmust úgy, hogy egy meghatározott *PMW* értéket elérő folyamatlépés éleit nem engedjük elhagyni, tehát azok minden esetben bekerülnek a modellbe. Az így definiált súlyokat a folyamatmodelleken reprezentatív módon megjeleníthetjük.

6.2 Súlyozás a folyamatlépések csoportosításában

Kutatásom során rájöttem arra, hogy az egyes folyamatlépések csoportosítása esetén az irodalomban található algoritmusok, csak nagyon kevés információt használnak fel, konkrétan csak a folyamatlépések sorrendiségét veszik figyelembe. Feltártam, hogy a folyamatmodellek csoportosítása esetén az egyes klaszterközéppontok definiálásakor más szempontok is fontosak lehetnek, a további folyamatlépések hovatartozása pedig szintén speciális értékektől függhet. Annak érdekében, hogy az egyes kitüntetett folyamatlépéseket kiemelhessük, súlyozási mértékekkel láthatjuk el azokat.

A súlyozásokat két féle szempont alapján vehetjük figyelembe a klaszterezés során:

- Minél nagyobb a súly a egy folyamatlépésnek, annál nagyobb valószínűséggel választjuk, mint klaszterközéppontot.
- Minél jobban közelíti egymást két folyamatlépés súlya, annál nagyobb eséllyel tartoznak egy csoportba.

Súlyozási módszerek a klaszterközéppont kiválasztásában

A klaszterközéppontok helyes definiálása egy lényeges lépése a csoportosítási eljárásoknak, így fontos, hogy a lehető legtöbb információt figyelembe vegyünk ezek kiválasztásánál. Az előzőekben bemutatott súlyozási értékekkel támogatva kiválaszthatjuk a „legsúlyosabb” folyamatlépéseket, mint csoportközéppontokat. Így alkalmazhatjuk az úgynevezett gravitációs csoportosítást, mely szerint a legsúlyosabb egyedek magukhoz vonzzák a kisebb egyedeket.

A módszer nagy előnye, hogy így a legnagyobb költségű, folyamatidejű vagy erőforrás igényű folyamatlépés lesz az, amelyik megmarad a tényleges modellben és ezzel vonjuk össze a kisebb esetünkben elhanyagolható folyamatlépéseket, ezzel is elősegítve a könnyebb átláthatóságot és elemezhetőséget.

Súlyozási módszerek a csoport-hovatartozás meghatározásában

A súlyozási módszer alkalmazásának a másik lehetősége a csoportosítások során, hogy az egy csoportba tartozó folyamatlépések hovatartozási-függvényét azok súlya definiálja.

Pontosabban a hovatartozási értékeket az egyes folyamatlépések három fő paramétere határozza meg: az azonos erőforrásokat használó folyamatlépések nagyobb eséllyel kerülnek egy csoportba, a magas költségű folyamatlépések szintén. Ezeknek a csoportoknak nem feltétlenül a folyamatmodellezésben van nagy hangsúlyuk, hiszen attól, hogy két folyamatlépésnek magas a költsége még egyáltalán nem biztos, hogy azok összevonhatók is, ennek a javasolt módszernek a folyamatok elemzésében van fontos szerepük. Továbbá az erőforrások alapján való csoportosításnak fontos szerepe lehet a folyamatmodellek strukturális optimalálásában is, mely szerint azokat a folyamatlépéseket melyekhez azonos erőforrást rendeltünk próbáljuk minél közelebb elhelyezni egymáshoz fizikailag, ezzel is gyorsítva az erőforrások elosztását és cseréjét.

7 Esettanulmányok

Dolgozatom végén három esettanulmányon keresztül igazolom a fejlesztett rendszer működését és felhasználhatóságát. Az első esettanulmány egy raktározási folyamat eseménynaplóit dolgozza fel a továbbfejlesztett *Alpha* algoritmussal. A második példában egy pénzügyi rendszereket jellemző folyamatmodellt, míg a harmadik példában egy tanulmányi előrehaladást vizsgáló folyamatot mutatok be.

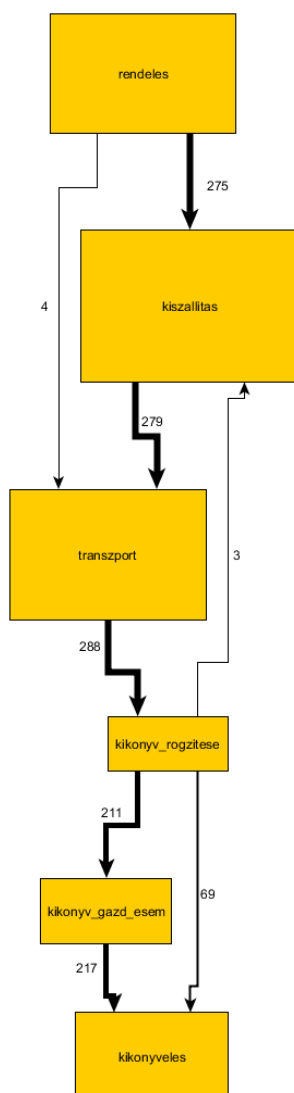
7.1 Alkalmazás logisztikai folyamat elemzésében

A rendszert a 6. táblázatban bemutatott eseménynapló adataival paramétereztem. A napló összesen 1990 db sort tartalmaz. Az adatokat SAP rendszerből származó log fájlból importáltam. A napló rögzítette az esetek sorszámát, a tevékenységek megnevezését és a bekövetkezésük idejét. A vizsgálat végső célja a beszerzési folyamatok elemzése volt.

6. táblázat - A vizsgált raktározási feladat eseménynaplójának részlete

ID	Tevékenység	starttime	endtime
100070628	10	2010-10-01 08:56:16	2010-10-01 08:56:16
100070764	10	2010-10-01 11:15:20	2010-10-01 11:15:20
100070888	10	2010-10-01 16:21:56	2010-10-01 16:21:56
100071030	10	2010-10-04 08:42:15	2010-10-04 08:42:15
100071055	10	2010-10-04 09:00:17	2010-10-04 09:00:17
100071354	10	2010-10-04 12:05:19	2010-10-04 12:05:19
100072225	10	2010-10-05 09:05:37	2010-10-05 09:05:37
100072232	10	2010-10-05 09:23:45	2010-10-05 09:23:45
100072326	10	2010-10-05 11:10:25	2010-10-05 11:10:25
100072410	10	2010-10-05 13:27:25	2010-10-05 13:27:25
100072514	10	2010-10-05 14:26:50	2010-10-05 14:26:50
100072530	10	2010-10-05 14:45:35	2010-10-05 14:45:35
100072631	10	2010-10-06 08:30:41	2010-10-06 08:30:41
100072971	10	2010-10-06 15:57:46	2010-10-06 15:57:46
100070628	11	2010-10-07 07:20:14	2010-10-07 07:20:14
100070764	11	2010-10-07 07:31:11	2010-10-07 07:31:11
100070628	12	2010-10-07 07:31:19	2010-10-07 11:54:07
100070764	12	2010-10-07 08:10:53	2010-10-07 11:00:23
100072631	11	2010-10-07 08:17:22	2010-10-07 08:17:22
100072631	12	2010-10-07 09:11:10	2010-10-08 13:31:58
100073004	10	2010-10-07 09:18:38	2010-10-07 09:18:38
100070628	15	2010-10-07 13:06:12	2010-10-07 13:06:12
100073205	10	2010-10-07 13:50:18	2010-10-07 13:50:18
100073216	10	2010-10-07 14:06:02	2010-10-07 14:06:02

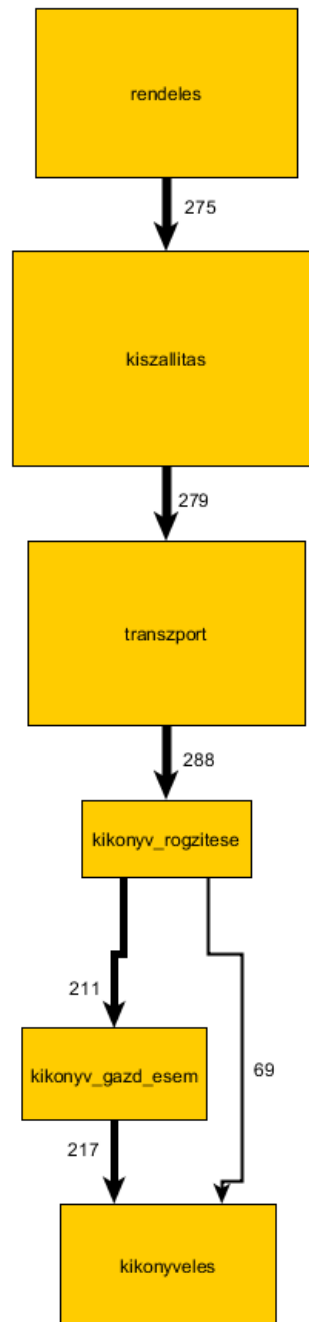
A továbbfejlesztett *Alpha* algoritmust 0,3-as határértékkel futtatva a 22. ábra által mutatott folyamatmodellt tártam fel. Az eseménynapló egy jól ismert folyamatmodellt ír le, így rendelkezésemre álltak olyan dokumentumok melyekkel összevettem a rendszer által adott modellt. Ezek alapján megállapítható, hogy a valósághoz közel álló folyamatmodellt tártam fel, azonban a modellben olyan ritka élek is megfigyelhetők, amelyek az eredeti dokumentációban nem szerepeltek. A modellen (22. ábra) már láthatóak a kidolgozott súlyozási módszerek, az élek súlya arányos azok vastagságával, míg a folyamatlépések súlya a jelölő négyzet nagyságával arányos.



22. ábra. A feltárt folyamatmodell

A 6. táblázatban definiált eseménynapló alapján feltárt modell.

Fontos, hogy a valóságban elhanyagolható élek is megjelentek a feltárt modellben, így az él-gyakorisági határértékeket hangolnom kellett, ebben az esetben 0,6-os határértékkel paraméterezve az algoritmus 23. ábra által mutatott modellt eredményezte.



23. ábra. A szigorúbb határérték eredménye

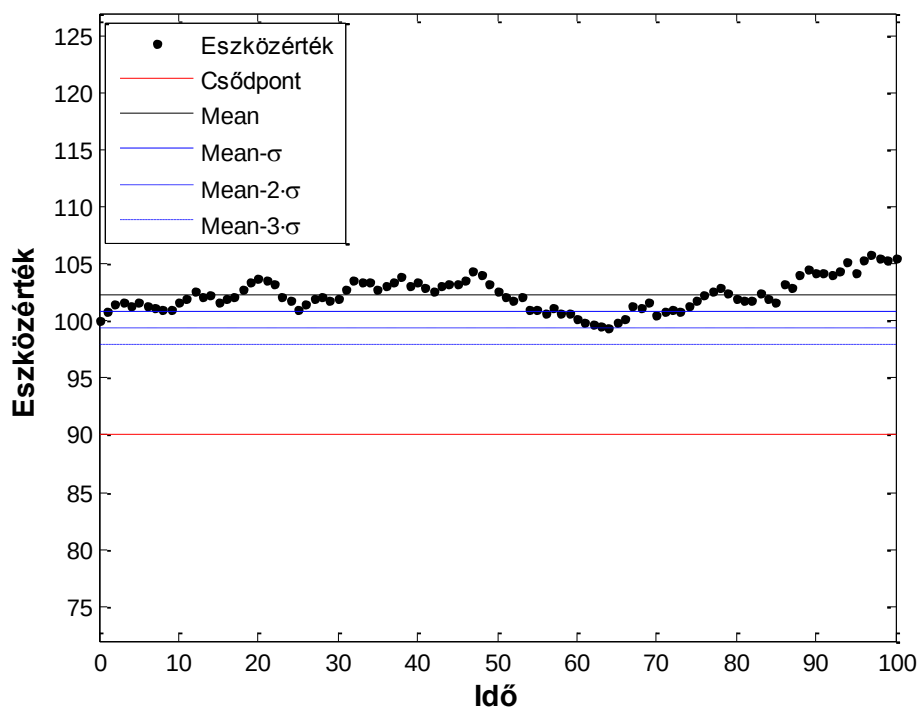
Összevetve a 22. ábra által mutatott modellel láthatjuk, hogy a kis gyakorisággal rendelkező éleket elhagytuk.

Ezen a kevésbé összetett, néhány folyamatlépésből álló példán is látható, hogy az általam javasolt és kifejlesztett módszer hatásos és célravezető lehet az *Alpha* algoritmus

esetén. Természetesen minél finomabban, pontosabban hangoljuk ezeket a paramétereket, annál pontosabb modellt kapunk eredményül.

7.2 Alkalmazás pénzügyi kockázatok feltárásában

A második esettanulmányban olyan pénzügyi folyamatot vizsgáltam, melyben rögzítve vannak az egyes pénzügyi események, és az is, hogy egy vállalat csődbe ment-e. Az eseménynaplót egy szimuláció generálta. Minden szimuláció (összesen 10.000) az alábbi (24. ábra) inicializációs szakasz egy lehetséges folytatását szimulálja. A vállalat csődbe megy, ha az eszközérték a csődpont alá kerül, ez az esemény az 5. *esemény* lesz. Kezdetben a cég biztonságban van, ezt az állapotot az eseménynaplóban 0 jelöli.

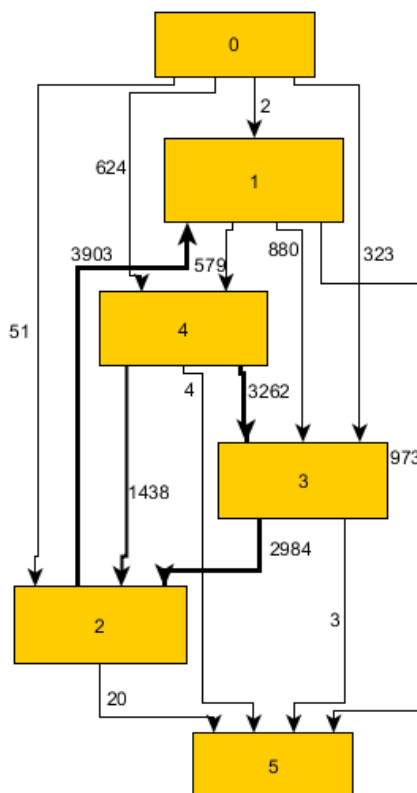


24. ábra. A vállalat eszközértéke

A szimuláció az eszközértéket szimulálta időegységenként.

Az előzőek értelmében minden szimuláció eseménynaplója a 0 eseménnyel kezdődik, és ha ezután bármely másik esemény bekövetkezik, akkor a folyamat már nem tér vissza a 0 állapotba. Minden szimuláció a csődig tart, tehát minden egyes szimulációban az 5-ös esemény pontosan egyszer szerepel, mely az utolsó esemény. A 0 és 5-ös közötti események a statisztikai folyamatirányításban alkalmazott szabályozókártyák alapján jelzik, hogy a csődvalószínűség (az 5-ös esemény kialakulásának valószínűsége) számottevő lett. Minden szimuláció során, minden időpontban, minden szabály teljesülése ellenőrzésre került.

Az így tenyésztett eseménynaplóra a fejlesztett keretrendszer a 25. ábra által mutatott folyamatmodellt eredményezte. A modellen jól látszik, hogy mely események után kerül csődbe a cég, ill. milyen események vezetnek ehhez. Az ábrán látható, hogy csak az élek gyakoriságát jelöltük különböző vastagságú vonallal, az egyes folyamatlépéseket ebben az esetben nem súlyoztuk, mivel ezeket azonos súlyúnak tekintettem. A modellt az *Alpha* algoritmussal tártam fel.



25. ábra. A csőd bekövetkezési folyamatmodell

A modellen látható, hogy a 0 biztonsági állapotból az 5-ös csőállapotba miként juthat el a cég.

Ezzel az esettanulmánnyal igyekszem rávilágítani arra, hogy a folyamatbányászati eszközöket nagyon széles területen lehet alkalmazni és elegendő a rendszer által rögzített eseménynapló ahhoz, hogy modellezni tudjuk ezeket a folyamatokat.

7.3 Tanulmányi előrehaladást leíró folyamatmodell identifikálása

Dolgozatom végén a *Heurisztikus bányászat* és a klaszterező algoritmusunk alkalmazhatóságát mutatom be a Pannon Egyetem gépészmérnöki alapképzésben tanuló hallgatók előmenetelének vizsgálata során. Az adatbázis (7. táblázat) a 2005. és 2013. között kezdett hallgatók tárgyteljesítését tartalmazza, természetesen a hallgatókat csak sorszám jelöli. Az eseménynapló tartalmazza továbbá a tárgy felvételének időpontját is (melyik félév).

7. táblázat - Gépészmérnök hallgatók tárgyfelvételi adatbázisának részlete

Tárgy	starttime	endtime
VEMKGEB122M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKMA1143V	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKMAB122M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKMAB112M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEGTGAB114M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKFI1312A	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKGEB112M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKGEB122M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKMA1143V	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKMAB122M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKGEB114A	2005-09-01 00:00:00	2006-01-12 00:00:00
VEGTGAB114M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKTT1312S	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKGEB143G	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKFI1312A	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKMAB112M	2005-09-01 00:00:00	2006-01-12 00:00:00
VEMKGEB112M	2006-02-01 00:00:00	2006-06-14 00:00:00
VEMKFOB333S	2006-02-01 00:00:00	2006-06-14 00:00:00
VEMKTN5344K	2006-02-01 00:00:00	2006-06-14 00:00:00
VEMKKVB212M	2006-02-01 00:00:00	2006-06-14 00:00:00
VEMKGEB213A	2006-02-01 00:00:00	2006-06-14 00:00:00
VEMKTT1312P	2006-02-01 00:00:00	2006-06-14 00:00:00
VEMKAKB212B	2006-02-01 00:00:00	2006-06-14 00:00:00
VEMKGEB112M	2006-09-01 00:00:00	2007-01-12 00:00:00

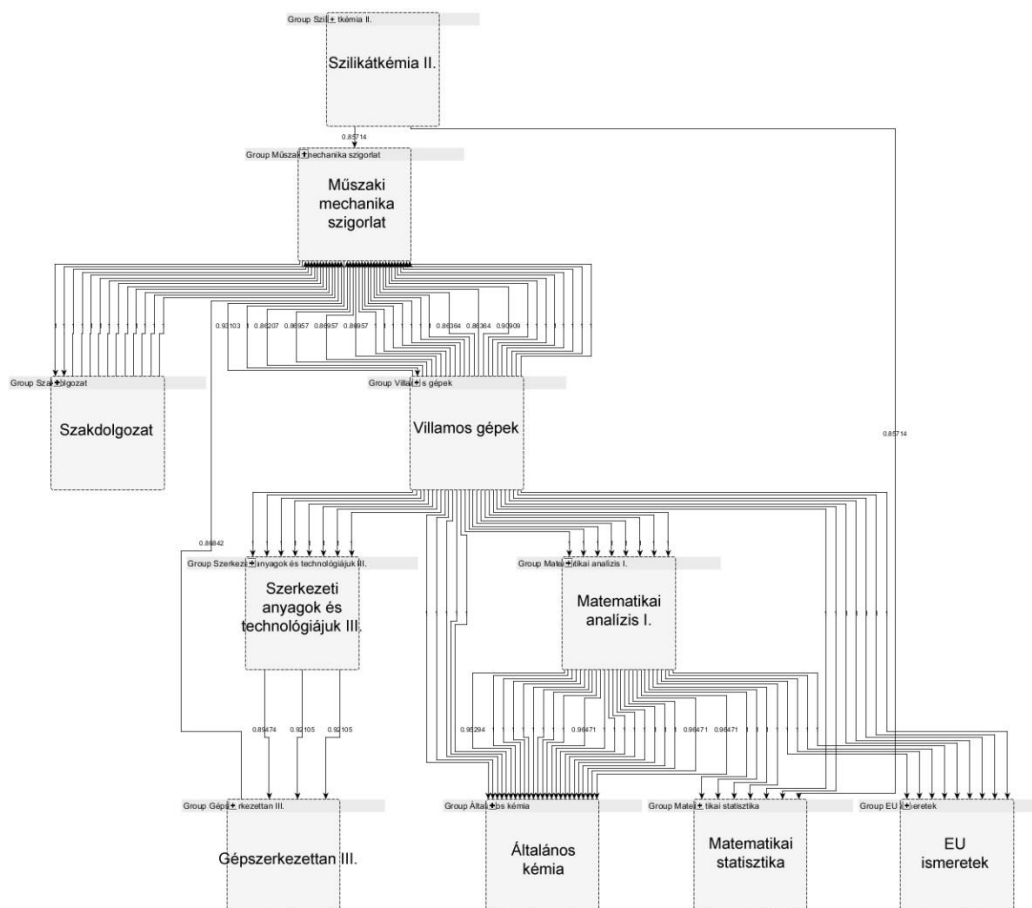
A folyamat feltárására az *Alpha* és a *Heurisztikus* algoritmusok önmagukban nem voltak alkalmazhatók, mert az egyes tárgyfelvetelek egy időben történtek, melyet az algoritmus még nem tud kezelni, ezért a naplót és az egyes tárgyak közötti kapcsolatot

egy egyedi programmal tártam fel. A feltárt kapcsolati mátrixra már alkalmazni tudtam a *Heurisztikus* algoritmust.

Az elkészült modellen alkalmaztam a fejlesztett *Fuzzy miner* klaszterezési eljárást, mellyel összesen 10 db klaszterre osztottuk a modellt. Ezen klasztereket a 26. ábra szemlélteti, ahol a klaszterek nevei az alábbiak:

- EU ismeretek
- Gépszerkezettan III.
- Matematika analízis I.
- Matematikai statisztika
- Műszaki mechanika szigorlat
- Szakdolgozat
- Szerkezeti anyagok és technológiájuk III.
- Szilikátkémia II.
- Villamos gépek
- Általános kémia

A klaszterezéseket megvizsgálva arra a következtetésre jutottam, hogy az algoritmus megfelelően működik, hiszen egymáshoz hasonló tárgyakat vont össze egy-egy csoportba és itt a hasonlóság alatt a tárgy-gráfban való elhelyezkedéseket tekintettük mérvadónak. A 26. ábra tökéletes példája annak, hogy a klaszterezési eljárások segítségével egy sokkal letisztultabb és átláthatóbb modellt kaphatunk folyamatainkról. Természetesen ezek a csoportok az yEd szoftver segítségével kibonthatók és a klaszterek összetettsége is elemezhető.



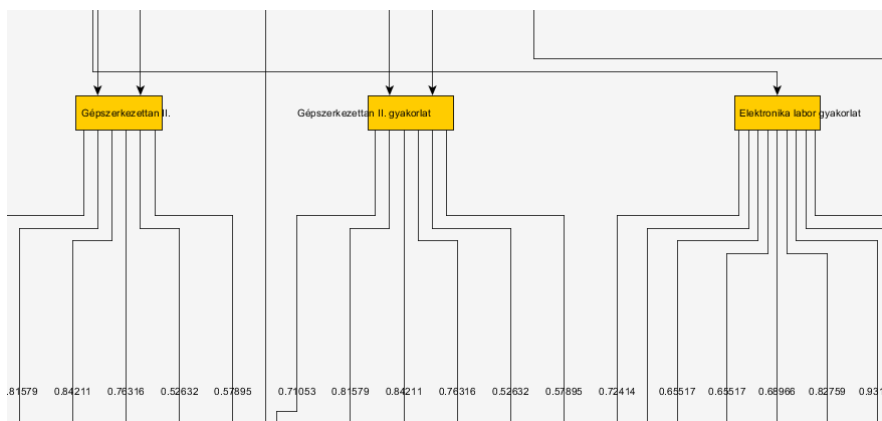
26. ábra. Gépészmérnöki tárgyak klaszterezése

A tantárgyteljesítések alapján a fenti 10 klaszterbe csoportosította a fejlesztett program a tárgyakat.

A 27. ábra a teljes modell egy részletét mutatja be, ahol az ábrázolt három folyamatlépés, vagyis tárgy (Gépszerkezettan II., Gépszerkezettan II. gyakorlat és Elektronika labor gyakorlat) egymással párhuzamos, ami logikus, hiszen ezek a tárgyak a tanterv szerint is azonos félévben szerepelnek.

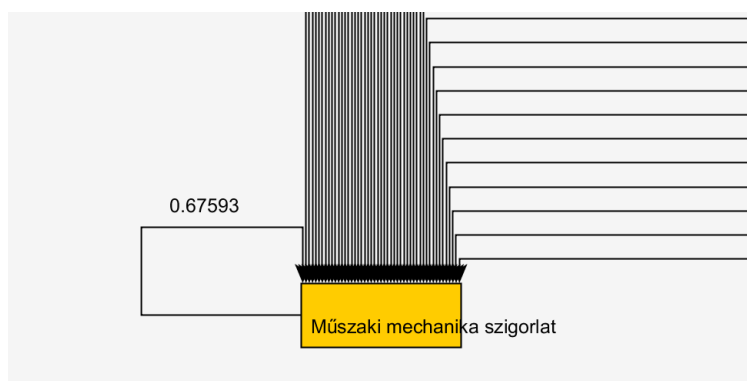
A 28. ábra a gépészmérnök hallgatók egyik legnépszerűbb tárgyát emeli ki, a műszaki mechanika szigorlatot. A modellnek ez az apró részlete nagyon reprezentatív, hiszen nemcsak jól ábrázolja, hogy rengeteg helyről érkeznek ide a hallgatók, hanem azt is, hogy az esetek több mint 67%-ban újrafelvételre is kerül.

A teljes folyamatmodell (29. ábra) túlságosan is összetett ahhoz, hogy jelen dolgozatban részletezni tudom, azonban ezen folyamatmodell alapján már alkothatunk olyan folyamat-szimulációt, amely segítségével részletesen elemezhetjük azt, és értékes információkat nyerhetünk a hallgatók tantárgyteljesítéséről és előmeneteléről.



27. ábra. Párhuzamos tárgyak megjelenése a modellben

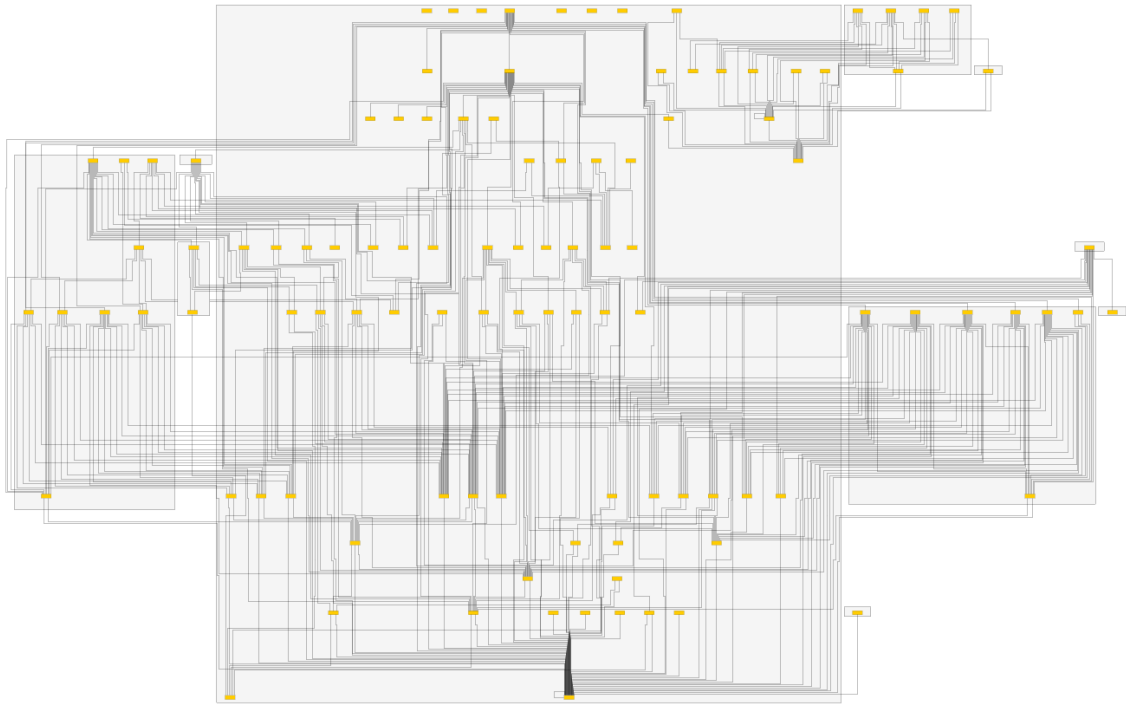
Az ábrán jól látható, hogy a három tárgy párhuzamos egymással, ami a valóságnak megfelel.



28. ábra. A Műszaki mechanika szigorlat a legnépszerűbb

A feltárt folyamatmodell legtöbb éllel rendelkező tárgya, továbbá a visszacsatolt élről leolvasható, hogy az esetek több mint 67%-ban újra felvételre kerül a tárgy.

Természetesen a folyamatbányászati eljárásokkal feltárt modelleknek nem pusztán a vizualizáció a céljuk. A feltárt modellek ismeretében olyan szimulációs modelleket készíthetünk, melyekkel teljes körű vizsgálatokat is végezhetünk ezeken a modelleken. Annak köszönhetően, hogy ezeket az algoritmusokat MATLAB környezetben fejlesztettem, elérhetővé váltak arra a célra, hogy bármely másik rendszerrel integrálhatóak legyenek.



29. ábra. A teljes gépészmérnöki folyamatmodell

Látható, hogy a teljes modell nagyon komplex, így az elemzésére folyamat-szimulációt javasolunk.

A fentiek tekintetében, az előzőekben bemutatott esettanulmány, mely a gépészmérnöki hallgatók tanulmányi előre menetelét tárja fel, egy olyan részletességű modellt kaptam, melyre folyamat-szimulációs eszközöket alkalmazva az adatokon túlélési elemzéseket is végezhetünk, mely azon túl, hogy feltárhatja a „kihalást” okozó elemeket, olyan információkkal is szolgálhat, amelyeket felülvizsgálva az oktatási színvonal javítható, esetleg optimalizálható.

8 Összefoglalás

Kutatásom tárgya a folyamatmodellek építése és csoportosítása folyamatbányászati eszközök alkalmazásával. Első lépésként megvizsgáltam, hogy a szervezetek által generált eseménynaplók alkalmasak-e automatikus folyamatmodell építésére. A folyamatbányászat eszközeinek segítségével a különböző eseménynaplók alapján feltárható egy rendszer (technológiai, logisztikai, üzleti) teljes folyamatmodellje és ezáltal elérhetővé válik a rendszer teljes értékű vizsgálata, elemzése.

A munkám legfontosabb eredménye az alap folyamatbányászati algoritmusok fejlesztése, hiszen ezek az algoritmusok eddig nem voltak elérhetőek, pusztán alkalmazni lehetett őket a piacon megtalálható szoftverek segítségével. Az algoritmusokról megjelent irodalmak alapján az algoritmusok matematikai háttéréről kaptunk információt, magukat az algoritmusokat MATLAB-ban fejlesztenem kellett. Kutatásom eredményeképpen ezek az algoritmusok elérhetővé váltak és ezáltal fejleszthetők, alkalmazhatók. További nagyon fontos eredményeim közé tartozik ezeknek az algoritmusoknak a továbbfejlesztése és gyorsítása.

A dolgozatomban bemutattam, hogy három folyamatbányászati algoritmus fejlesztésén túl, fejlesztettem egy *xgml* fájlokat generáló programot, melynek segítségével a feltárt modelleket az yEd programmal jeleníthetjük meg. Rámutattam, hogy a további fejlesztési lehetőségek közül az általam kidolgozott súlyozásos módszer segítségével elérhetővé válik a folyamatlépések megfelelő csoportosítása, mely módszerek felhasználásával közelíthetjük a valós folyamatokat, de közben a modell áttekinthetőségét is megőrizzük.

A dolgozatom zárásaként három esettanulmányt mutattam be. Az első egy ipari példa volt, mellyel bemutattam az *Alpha* algoritmus hatékonyságát, míg a másodiknál pénzügyi kockázat elemzési folyamatot vizsgáltam. A harmadik esettanulmányomban egy egyetemhez kötődő tanulmányi előrehaladási példán illusztráltam a csoportosítási algoritmus hatékonyságát.

A fejlesztett algoritmusokat MATLAB programcsomagban valósítottam meg, ezáltal elérhetővé vált az eljárás integrálása a MATLAB széleskörű eszköztárával és elérhetővé vált e folyamatbányászati algoritmusok további fejlesztése is.

A folyamatbányászati algoritmusok kutatásában még komoly potenciált látok, így a munkámat a jövőben is folytatom. Munkám következő lépéseként a szakirodalomban

található algoritmusok egyik legnagyobb hátrányát fogom kiküszöbölni, mely szerint a rendszereket Petri-hálóként reprezentálja. A probléma az, hogy ezek a modellek nem veszik figyelembe az időbeli párhuzamosságot. Ezeknek az időbélyegeknak a kezelése fontos feladat, hiszen az eseménynaplókban, ha két esemény mindig egy időpontban következik be, akkor azok sorrendje esetleges, így sem az *Alpha*, sem a *Heurisztikus* algoritmusok nem képesek feltárni ezeket.

A folyamatbányászati algoritmusok fejlesztése a korábbi kutatásaim során merült fel, így a legfontosabb célom, hogy ezekkel folyamat-szimulációs modelleket készítsünk. Erre a célra a SIMUL8 szoftvercsomagot javasolok, így a fejlesztések során a mostani paraméterek mellett, további információkat is ki kell nyernünk ahhoz, hogy automatikusan folyamat-szimulációs modelleket tudjunk generálni. Ehhez szükségem lesz a mostani „*xgml*” fájlokat generáló programot úgy továbbfejleszteni, hogy SIMUL8 által is kezelt „*xml*” fájlokat tárhassak fel.

Bízom abban, hogy a létrehozott eszköztárat, melynek angol nyelvű kézikönyvének ismertetésére a dolgozat mellékletében kerül sor, minél szélesebb körben fogják alkalmazni.

9 Irodalomjegyzék

- [1] W. M. P. van der Aalst, A. H. M. ter Hofstede, Weske M.: *Business Process Management: A Survey*, Business process management, 1-12, Springer, 2003
- [2] T. Ruppert, G. Belvárdi, Z. Gyozsán, J. Abonyi, Zs. Ulbert: Logisztikai tevékenységet támogató folyamatmodellek fejlesztése és Monte-Carlo szimulációval történő kialakítása, Műszaki Kémiai Napok 2012, 47-53. 2012
- [3] Zs. Ulbert, T. Ruppert, G. Belvárdi, Z. Gyozsán, J. Abonyi: *Interactive Balanced Scorecard based on Simulation of Production and Services Systems*, Factory Automation 2012, 88-91. 2012
- [4] W.M.P. van der Aalst: *Business process mining: An industrial application*, Information Systems, 713-732, Elsevier 2007
- [5] Wil M.P. van der Aalst: *Process Mining*, Book, Springer, pp. 370, 2011
- [6] www.processmining.org adatai alapján (2014.01.18.)
- [7] A.K.A de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters: *Process Mining: Extending the α -algorithm to Mine Short Loops*, Eindhoven University of Technology, Eindhoven, 2004
- [8] A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros: *Process Mining with the Heuristics Miner Algorithm*, Technische Universiteit Eindhoven, 1-34, 2006
- [9] A.K. Alves de Medeiros, A.J.M.M. Weijters and W.M.P. van der Aalst: *Using Genetic Algorithms to Mine Process Models: Representation, Operators and Results*, Research School for Operations Management and Logistics, 2005
- [10] Dolean C-C, Petrusel R.: *A Mining Algorithm for Extracting Decision Process Data Models*, Informatica Economica, 79-95, Academy of Economic Studies-Bucharest, 2011
- [11] Werner-Stark A, Dulai T. Agent-based analysis and detection of functional faults of vehicle industry processes: a process mining approach In: Jezic G, Kusek M, Nguyen N T, Howlett R J, 424-433, 2012
- [12] Werner-Stark Á, Németh E, Gerzson M, Hangos K M *Model-based fault detection and isolation using process mining*, 851-856, 2011

- [13] B.F. van Dongen, A. Adriansyah: *Process Mining: Fuzzy Clustering and Performance Visualization*, Business Process Management Workshop, 158-169, Springer, 2010
- [14] Günther C. W., W. M.P. van der Aalst: *Fuzzy Mining – Adaptive Process Simplification Based on Multi-Perspective Metrics*, Business Process Management, 328-343, Springer, 2007
- [15] <https://fluxicon.com/disco/> (2014.03.18)
- [16] <http://www.perceptivesoftware.com/products/perceptive-process> (2014.03.18)
- [17] <http://fluxicon.com/nitro/> (2014.03.18)
- [18] Werner Á.: *Folyamatbányászat* (slides)
- [19] I.E.A. Segers: Investigating the application of process mining for auditing purposes; Deloitte Enterprise Risk Services; PhD thesis, 2007
- [20] J. Dombi: *Intelligens rendszerek és alkalmazásaik* (notes), 2009
- [21] <http://www.yworks.com/> (2014.11.03.)
- [22] Wil M.P. van der Aalst and K.M. van Hee.: *Workflow Management: Models, Methods, and Systems*, MIT Press, 2004.
- [23] Wil M.P. van der Aalst: *Workflow Mining: Discovering process models from event logs*, IEEE Transaction on Knowledge and Data Engineering, vol. 16, no. 9, pp. 1128-1142, 2004.