Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Federált Bayes-háló tanulás neonatális intenzív centrumok elosztott adataiból

TDK dolgozat

Készítette:

Kisida Júlia Lili

Konzulens:

dr. Antal Péter
dr. Szabó Miklós

2023

Tartalomjegyzék

Kivonat

Az értelmezhető és megbízható mesterséges intelligencia módszerek iránti igény miatt és az okozatiság közvetlen reprezentálásának a lehetősége miatt egyre nő az igény a Bayes-hálózatok hatékony, felskálázható gépi tanulására. A struktúra tanulásának diszkrét jellege, alapvető megfogalmazásban NP-teljes volta és nehéz párhuzamosíthatósága miatt ez hosszú időn át nem tűnt feloldhatónak. Egy 2018-ban publikált újszerű megközelítés [16] egy folytonos optimalizációs problémaként fogalmazta újra a feladatot az építendő irányított körmentes gráf aciklikusságát biztosító feltétel átdolgozásával. Ezáltal a feladat már megoldható standard numerikus algoritmusokkal, ilyen például a kiterjesztett Lagrange-típusú optimalizációs módszer.

Az adatok elosztott volta és személyi, illetve intézményi szintű védettsége további kihívást jelent a gépi tanulási eljárások alkalmazására. A federált tanulási módszerek erre jelentenek megoldást, mivel ekkor a partnereknek nem szükséges az adatokat, elegendő a modellparamétereket megosztani a központtal. Az algoritmus egyes részei így lokálisan, a partnereknél futnak, majd a modellparaméterek megosztása után a központi globális modellen kerülnek végrehajtásra a szükséges lépések. A federált tanulási keretrendszerbe a Bayes-hálózatok új, folytonos alapú optimalizációja természetes módon tagolható be.

A dolgozatomban a bemutatom a NOTEARS és a federált NOTEARS-ADMM [12] algoritmusokat, melyek folytonos alapú optimalizációs problémaként tekintenek a Bayes-háló tanulásra. Szisztematikusan kiértékelem a teljesítményüket különböző hiperparaméter beállítások mellett és javaslatot teszek egy federált tanulási protokollra, a hiperparaméterek megválasztására. A generált gráfok elemzését több szempont szerint is elvégzem: egyrészt vizsgálom a federált algoritmus eredményét különböző mintaszámú kliensek esetén, másrészt külön figyelmet fordítok a partnerek adataiból lokális, globális és federált módon épített gráfok különbségeinek feltárására.

A Bayes-hálózatok federált tanulását neonatális intenzív centrumok koraszülött intenzív ellátását leíró adatokon végzem el a centrumok által 2005 és 2013 között rögzített adatokat felhasználva. A területen a kezelést leíró döntési protokollnak nagy súlya van, hiszen újszülöttek élete a tét, ezért nagyon fontos, hogy az egyes döntési stratégiák elemzése megtörténjen és lehetőség legyen a visszacsatolásra. A mesterséges intelligencia bevonása ugyanakkor nemcsak erre teremt lehetőséget: az adatból épített oksági struktúrák vizsgálata akár eddig nem ismert összefüggésekre is fényt deríthet.

Abstract

Due to the demand for explainable and trustworthy artificial intelligence methods, as well as the possibility of representing causality, there is a growing need for Bayesian networks' scalable and efficient machine learning methods. The discrete nature of structural learning, its NP-completeness, and its difficulty in parallelization have made it seem unsolvable for a long time. A novel approach published in 2018 [16] reframed the task as a continuous optimization problem by reformulating the condition ensuring the acyclicity of the directed acyclic graph to be constructed. This made the task solvable with standard numerical algorithms, such as the augmented Lagrangian method.

The distributed nature of data and the personal and institutional privacy further challenge the application of machine learning methods. Federated learning methods provide a solution for this issue: clients do not need to share their data; it is sufficient to only share the model parameters with the central server. Thus, individual parts of the algorithm run locally on clients, and after sharing the model parameters, the central global model executes other necessary steps. The new continuous optimization of the learning of Bayesian networks can naturally fit into the federated learning framework.

In my paper, I summarize the NOTEARS and the federated NOTEARS-ADMM [12] algorithms that are regarding the learning of Bayesian networks as a continuous optimization problem. I systematically evaluate their performance under various hyperparameter settings and propose a federated learning protocol and hyperparameter selection. I analyze the generated graphs from multiple perspectives: first, I examine the results of the federated algorithm for different numbers of clients and different sample sizes. I also pay special attention to uncovering the differences between graphs built locally, globally, and in a federated setting.

For the learning of Bayesian networks, I use the data of the Neonatal Intensive Care Units, recorded between 2005 and 2013. In the field, decision protocols describing treatment are of great significance since the lives of newborns are at stake. The analysis of different decision strategies is crucial because it can make feedback possible. The involvement of artificial intelligence, however, not only enables this but also allows for the investigation of causal structures built from data, potentially revealing previously unknown correlations.

1. fejezet

Bevezetés

Alapvető emberi sajátosságunk, hogy oksági összefüggésekben szeretünk gondolkozni. Egy jelenség vizsgálatánál szeretjük feltenni a *miért?* kérdést, módszereket keresünk arra, hogy megtaláljuk bizonyos jelenségek okát vagy bizonyítsuk az oksági kapcsolatot két jelenség között. A tudomány egyik legalapvetőbb célja az oksági kapcsolatok természetének megértése [15], ok és okozat azonosítása. Ma már képesek vagyunk nagy mennyiségű adat rögzítésére, ám az ezek közti összefüggések feltárása továbbra is nagy kihívás, amelyben az utóbbi néhány évtizedben jelentek meg csupán az első elméleti és gyakorlati eredmények a mesterséges intelligencia területén.

Számos tudományos kutatás esetén, amely oksági összefüggések feltárására irányul, felmerülhet a kísérletbe való beavatkozás kérdése [15]. Azonban ez sok esetben etikai okok miatt nem valósítható meg. A különböző mesterséges intelligencia módszerek bevonásával azonban újfajta lehetőségek nyílnak az oksági adatelemzés területén: bizonyos oksági kapcsolatok azonosíthatóak és jellemezhetőek pusztán megfigyelt adat alapján is, beavatkozás, manipuláció nélkül.

A Bayes-hálók modellosztályban a valószínűségi változók közötti közvetlen okozatiság reprezentálható körmentes, irányított gráfok, DAG struktúrák által. A módszernek számos alkalmazási területe említhető: a döntéstámogatás, orvosi képalkotás, de természettudományi és biológiai területén is nagy jelentősége van [15].

A Bayes-hálók adatból való hatékony, felskálázható, robusztus tanulása azonban nehéz feladatnak bizonyult, amely különös kontrasztot jelentett az utóbbi évtized gradiens alapú optimalizálás sikerei, illetve a mesterséges általános intelligenciát célzó nagy nyelvi modellek megjelenése mellett. Ennek áthidalására jelentek meg a Bayes-hálók tanulásán belül a kombinatorikus megközelítés mellett a folytonos optimalizációt használó módszerek [15]. Bár a struktúrák tanulása NP-nehéz probléma [9], ami főként a kombinatorikus aciklikussági kényszer miatt van így [16]; azonban egy új, aciklikusságot biztosító feltétel bevezetése új közelítő megoldásokat kínál a problémához. A folytonos optimalizációt használó módszerek jelenleg is kiemelten kutatottak, az első ilyen a 2018-ban publikált *DAGs with NO TEARS: Continuous Optimization for Structure Learning* című tanulmányban [16] bemutatott NOTEARS algoritmus volt, mely megjelenése óta nagy népszerűségnek örvend, számos bővített és továbbfejlesztett változata jelent meg: ilyen például a NOTEARS-ADMM [12] vagy a WITHTEAR algoritmus [7].

Egy másik népszerű kutatási terület valamennyi gépi tanulási módszer, így a Bayes-háló tanulás területén is az adatok védettségét megőrző tanulási módszerek, amely az MI új szabályozásaiban egy központi elem, mind az EU AI Act készülő szabályozásában [14] vagy az éppen megjelent USA-beli szabályozásban [13]. Ekkor anélkül szeretnénk valamilyen érzékeny, elosztott adatból oksági struktúrát tanulni, hogy a partnereknek meg kellene osztani az adataikat egymással vagy egy központi szerverrel. Erre a federált tanulási módszerek nyújthatnak megoldást, amelyek általában úgy működnek, hogy a partnereknél,

lokálisan fut az algoritmus egyik része, míg más lépések egy központi szerveren kerülnek végrehajtásra és általában elegendő, ha az egyes partnerek elküldik a modellparamétereket a központi szervernek, ahol ezek ismeretében frissíthetők a globális modellparaméterek, amiket a szerver ismét szétküldhet az egyes partnereknek [4, 3]. Általában ezek a módszerek sem küszöbölik ki teljes mértékben az adatcserét, sokszor a mintaszám és általános statisztkai paraméterek (átlag, szórás) megosztására szükség van. Dolgozatomban a federált tanulás statisztikai és gépi tanulási aspektusát vizsgálom és nem tárgyalom az adatvédelmi és adatbiztonsági kérdéseket.

A NOTEARS-ADMM algoritmus [12] a NOTEARS algoritmus egy federált változata. Az ADMM algoritmus rendelkezik mind a kiterjesztett Lagrange-típusú módszer előnyös konvergencia tulajdonságával (egy új tag bevezetésével enyhébb feltételek megléte mellett is differenciálható marad a duális optimalizálási probléma), mind a duális optimalizálás szeparálhatóságával [5], ezért a segítségével lehetséges részekre bontani a DAG-tanulási problémát, megőrizve a kiterjesztett Lagrange-típusú módszer előnyeit. Mint a federált tanulási módszereknél megszokott, itt is vannak a tanulásnak a partnernél futó, lokális lépései, valamint a központi szerveren futó golobális lépések, melyek a partnerek által megosztott modellparamétereket használják.

A magyar egészségügyben számos területen megvalósul már az adatgyűjtés, azonban az adatok kezelése jellemzően központilag történik. Ilyen a neonatális centrumokban 2002 óta megvalósuló adatgyűjtés is. Az országban 22 neonatális intenzív centrum működik, ahol koraszülöttek intenzív ellátását végzik. Ez együtt jár sok olyan helyzettel, ahol azonnali döntésre van szükség. Nagyon fontos lenne, hogy az adatok alapján lehetőség legyen visszajelzésre, ami alapján az egyes centrumok felül tudják vizsgálni az alkalmazott kezelési módokat, döntési protokollokat. Mivel az adatbázisban több száz változóról van adat rögzítve, a hagyományos statisztikai módszerek csak bizonyos mértékig hívhatók segítségül. A mesterséges intelligencia bevonásával azonban egy újfajta szemléletmód érvényesülhet, hiszen alkalmazásával hatékony módon lehet feltárni az adatban jelentkező strukturális különbségeket, rejtett összefüggéseket.

Ezen felül, federált adatelemző módszerek segítségével lehetőség nyílna a NIC (Neonatális Intenzív Centrum) adatbázis esetén is arra, hogy az adatokat védetten, elosztva lehessen tárolni. Ez nemcsak amiatt fontos, hogy az érzékeny adatoknak nagyobb fokú védettségét biztosítsunk: praktikus előnyei is vannak, hiszen az adatok kezelése egyszerűbbé, jobban átláthatóvá válik, ha az illetékes centrum lokálisan felel értük.

A dolgozatomban átfogó módon vizsgálom az oksági Bayes-háló tanuló módszerek alkalmazásának lehetőségét a magyar koraszülött (NIC) adatbázisban: foglalkozom az adatok előkészítésével, hiányzó adatok kipótlásával. Vizsgálom a NOTEARS és a federált változatának, a NOTEARS-ADMM algoritmusnak az eredményeit az adathalmazon. Javaslatot teszek egy átfogó federált keretrendszerre, amely (1) kezeli az adatok hiányos voltát, (2) felhasználja a folytonos DAG optimalizációt megvalósító NOTEARS algoritmust, (3) annak federált NOTEARS-ADMM kiterjesztését, és (4) 'bootstrap' újramintavételezési módszer használatát a struktúra tanulás bizonytalanságának kezelésére. Szisztematikus vizsgálatokat javaslok a teljes keretrendszer hiperparamétereinek a megválasztására, amelyet a NIC valós alkalmazási területén mutatok be. Bemutatom továbbá az algoritmusok érdekesebb futási eredményét is, azonban fontos rögzítenem, hogy jelen dolgozatnak nem célja az eredmények részletes orvosi értelmezése.

A saját kontribúcióm, a munkám újdonsága a NIC-re szabott keretrendszerben és az algoritmusok specifikus hangolásában rejlik. Az általam írt keretrendszer képes a NIC hiányos adatainak kipótlására, majd a kipótlás robusztusságának bizonyítására. Ezen felül tudja kezelni a NIC adatait, lehetőség van a NOTEARS és a NOTEARS-ADMM algoritmusok futtatására lokális, globális és federált beállítások mentén, illetve újramintavételezés esetén is, majd ezek összehasonlítására, teljesítményének mérésére valamilyen

objektív, számszerűsíthető metrika kidolgozása által. Saját kontribúcióm még az algoritmusok paramétereinek NIC-specifikus beállítása, illetve erre egy protokoll kidolgozása.

A dolgozatom a bevezetőt követően három nagyobb egységre bomlik: A második fejezetben a szükséges háttérismereteket mutatom be. Feltételezem, hogy az olvasó tisztában van a Bayes-hálók alapvető fogalmaival, ezért a NOTEARS, a NOTEARS-ADMM bemutatására fordítok több időt. Azon belül is bemutatom a DAG-tanulás területén megjelenő folytonos optimalizácót használó módszerek újítását, bizonyítom a módszer helyességét. Az algoritmus paraméterezéssel kapcsolatos elméleti részeire is külön figyelmet fordítok, ez a NOTEARS esetén a kiterjesztett Lagrange-típusú optimaliációs módszer, a NOTEARS-ADMM esetén pedig az ADMM algoritmus. A harmadik fejezetben bemutatom, milyen előkészületeket hajtottam végre a NIC-ben megvalósuló adatelemzés elősegítése végett: részletesen bemutatom, hogyan történt a hiányos adatok pótlása, és mindkét algoritmus esetén részletesen bemutatom a megfelelő paraméterek beállításának és megtalálásának folyamatát. A negyedik fejezetben a tényleges adatelemzési lépéseket foglalom össze, a célkitűzéseimnek megfelelően az eredmények módszertani aspektusaira fókuszálva, demonstrálom az algoritmusok teljesítményeit az adatbázisoban, végül értékelem az eredményeket, javaslatot teszek a keretrendszer továbbfejlesztésére.

2. fejezet

Adatok és módszerek

Ebben a fejezetben bemutatom a dolgozatomban előkerülő fogalmakat, módszereket és algoritmusokat, különös figyelmet fordítva utóbbiak újításaira a Bayes-háló tanulás területén.

2.1. A Bayes-hálókkal kapcsolatos alapfogalmak

Az alábbiakban bemutatom a DAG tanulással kapcsolatos legfontosabb alapfogalmakat. A bemutatott definíciók az alábbi források alapján készültek: [2, 15, 8], további részletek találhatók az eredeti tanulmányokban és tankönyvekben. (Az egyes definíciók esetén egyénként is megjelöltem azok forrását.)

Tekintsük $G = (V, E)$ irányított, körmentes gráfot, ahol $V = \{V_1, V_2, ...V_p\}$ a gráf csúcsai, melyek valószínűségi változókat reprezentálnak, $E \subseteq V \times V$ pedig a gráf élei, melyek a közöttük lévő kapcsolatoknak feleltethetők meg szemléletesen.

Az alábbi definíciók foglalják össze $X_1, X_2, ...X_n$ valószínűségi változók $p(X_1, X_2, ...X_n)$ együttes eloszlása és G DAG reprezentációja közötti kapcsolatot:

Definíció 1 (d-elválasztottság). [8] Legyenek $X, Y, Z \subset V$ a csúcsok diszjunkt halmazai G gráfban. X és Y d-elválasztottak Z által, azaz $X \perp\!\!\!\perp_d Y|Z$ pontosan akkor, ha minden közöttük vezető U útra igaz, hogy

- létezik egy $V_i \to V_k \to V_j$ vagy egy $V_i \leftarrow V_k \to V_j$ szakasz, ahol V_k Z-beli csúcs
- vagy létezik egy $V_i \to V_k \leftarrow V_j$ szakasz U-ban, ahol se V_k, se V_k leszármazottai nem Z-beli csúcsok ▪

Definíció 2. [2] A $p(X_1, X_2, ...X_n)$ eloszlás faktorizálható a G DAG szerint,ha

$$p(X_1, X_2, ...X_n) = \prod_{i=1}^{n} p(X_i|Pa(X_i)) \tag{2.1}$$

ahol $Pa(X_i)$ az X_i csomópont szülőinek halmaza G-ben. ▪

Definíció 3. [2] A $p(X_1, X_2, ...X_n)$ eloszlásra teljesül a globális Markov-feltétel G szerint, ha

$$\forall X, Y, Z \subseteq V : X \perp\!\!\!\perp_d Y|Z \Rightarrow X \perp\!\!\!\perp_p Y|Z \tag{2.2}$$

▪

A 3-mas definíció szerint tehát a G-ben jelen lévő d-elválasztott csúcsok az eloszlásban feltételes függetlenség formájában jelennek meg.

Definíció 4. [2, 15] G_1 és G_2 DAG gráfok pontosan akkor tartoznak egy ekvivalencia-osztályba, ha irányítatlan vázuk izomorf és csak ugyanazokat a v-struktúrákat tartalmazzák. ▪

Megjegyzés 1 *V-struktúrának nevezzük, amikor két nem szomszédos csúcsból egyaránt mutat él egy harmadik, leszármazott csúcsba.*

2.2. A NOTEARS algoritmus

Keressük azt a $G = (V, E)$ gráfot, amely az $\mathbf{X} = (X_1, X_2, ...X_d)$ valószínűségi vektorváltozó együttes $p(X_1, X_2, ...X_d)$ eloszlását reprezentálja, ahol $\mathbf{X} \in \mathbb{R}^{n\times d}$, n a teljes adat mintaszáma, d pedig G-ben lévő csúcsok száma.

DAG struktúra tanulása adatból NP nehéz feladat [9], főként a tanult DAG körmentességét biztosító feltétel miatt. A probléma alábbi, hagyományos megfogalmazása egy kombinatorikus feltétel bevezetését igényli, amely a tanult gráf körmentességét biztosítja:

$$\begin{aligned} &\min_{W\in\mathbb{R}^{d\times d}} F(W) \\ &\text{feltéve, hogy } G(W) \in \mathbb{D} \end{aligned} \tag{2.3}$$

A 2.3-mas egyenletben $G(W)$ a W szomszédsági mátrixból képzett gráf, $\mathbb{D}$ az irányított körmentes gráfok tere, $F : \mathbb{R}^{d\times d} \to \mathbb{R}$ pedig egy pontszám alapú függvény. Ebben a formában azonban az aciklikussági feltételt nagyon nehéz hatékonyan érvényre juttatni. A 2018-ban megjelent *DAGs with NO TEARS: Continuous Optimization for Structure Learning* [16] című tanulmány újításának köszönhetően a probléma átfogalmazása megtörténhetett: egy $h : \mathbb{R}^{d\times d} \to \mathbb{R}$ függvény bevezetésével a 2.4 már egy folytonos optimalizációs probléma:

$$\begin{aligned} &\min_{W\in\mathbb{R}^{d\times d}} F(W) \\ &\text{feltéve, hogy } h(W) = 0 \end{aligned} \tag{2.4}$$

A h függvény egy folytonos, deriválható függvény, mely pontosan akkor 0, ha argumentuma egy aciliklikus gráf.

A 2.4-es probléma így már standard numerikus optimalizációs algoritmusok alkalmazásával megoldható, ilyen például a kiterjesztett Lagrange-típusú módszer is.

Fontos kiemelni, hogy az algoritmus futtatása során az élek irányításáról csak annyit tehetünk fel, hogy az eredménygráf a megfelelő ekvivalencia-osztályba fog tartozini. Az élek irányítottságát csak ennek a tudatában érdemes vizsgálni.

Az alábbiakban részletesen kifejtem az algoritmus egyes részeinek működését, vizsgálom a h függvényt, bizonyítom, hogy valóban akkor és csak akkor 0, ha argumentumában egy aciklikus gráf szerepel. Kitérek a kiterjesztett Lagrange-típusú optimalizálási módszerre is, végül pedig az algoritmus pszeudokódjának megadásával szemléltetem az algoritmus működését.

2.2.1. Legkisebb négyzetek módszere

A pontszám alapú módszerek DAG-tanulás esetén azt használják ki, hogy a pontszám alapú függvény minimumának megtalálásával az adat valódi struktúrája kapható. A legkisebb négyzetek módszere is ilyen (az angol nyelvű szakirodalomban *least-squares loss*). Lényege az, hogy az eltérések négyzetösszegének minimumát megtaláljuk. A NOTEARS algoritmus is ezt használja:

$$F(W) = \ell(W; X) + \lambda\|W\|_1 = \frac{1}{2n}\|X - XW\|_F^2 + \lambda\|W\|_1 \tag{2.5}$$

ahol $\|W\|_1$ W 1-es normája, $\|X - XW\|_F^2$ pedig a Frobenius-norma. A függvényben a $\lambda\|W\|_1$ tag az L1-regularizáció.

2.2.2. Az aciklikusságot biztosító feltétel

Keressük azt a $h : \mathbb{R}^{d\times d} \to \mathbb{R}$ függvényt, amelyre igaz, hogy $h(W) = 0$ pontosan akkor, ha a W szomszédsági mátrix által reprezentált gráf DAG. Azt is szeretnénk továbbá, hogy h rendelkezzen az alábbi tulajdonságokkal [16]:

- $h(W) = 0$ akkor és csak akkor, ha W körmentes
- $h(W)$ értéke mutassa meg, hogy a W mátrix által reprezentált gráf 'mennyire DAG'
- h legyen sima
- h-t és a gradiensét legyen könnyű kiszámítani

h konstrukciójának megértéséhez először be kell látnunk néhány összefüggést. Ezek a tételek részletes bizonyításukkal mind szerepelnek az algoritmust bemutató tanulmányban [16], én a fontosabbakat emelem ki és foglalom össze.

Tétel 1. Egy bináris B mátrix $B \in \{0, 1\}^{d\times d}$ akkor és csak akkor DAG, ha

$$tre^B = d. \tag{2.6}$$

▪

Bizonyítás. B mátrix exponense az alábbi alakban írható fel:

$$e^B = \sum_{k=0}^{\infty} \frac{B^k}{k!} \tag{2.7}$$

k=0 esetben $B^0 = I$, ahol I a $d \times d$-s egységmátrix. Ekkor B mátrixtól függetlenül teljesül, hogy $trB^0 = trI = d$, hiszen egy $d \times d$-s egységmátrix nyoma pontosan d.

k=1 esetben $B^1 = B$, amire $trB = 0$ akkor és csak akkor, ha B mátrixban nincsenek hurokélek, hiszen pontosan ekkor lesznek a főátlóban lévő elemek egyenlők 0-val.

k=2 esetben vegyük észre, hogy a B_{ii}^2 azoknak a kettő élhosszúságú irányított utaknak a száma, melyek az i-edik csúcsból indulnak és az i-edik csúcsban érnek véget. Tehát ez az i-edik csúcsot tartalmazó, kettő élhosszúságú irányított körök száma a gráfban. Ha a mátrix nyomát képezzük, azaz vesszük a $B_{11}^2 + B_{22}^2 + ... + B_{dd}^2$ összeget, az csak akkor lehet nulla, ha egyetlen irányított kör sincs a gráfban (mivel nincsenek negatív élsúlyok).

Ezt folytatva minden B^k mátrixra igaz, hogy a nyoma akkor és csak akkor 0, ha nincs a gráfban k élhosszúságú irányított kör. Mivel $k = 0$-ra $trB^0 = d$ összefüggés állt fenn, ezért a $k = 0$-tól ∞-ig, az $\sum_{k=0}^{\infty} \frac{B^k}{k!}$ összeg csak akkor lehet nulla, ha semmilyen $k > 0$-ra nincs k hosszúságú irányított kör a gráfban, azaz ha a gráf DAG.

Tétel 2. Egy W mátrix $W \in \mathbb{R}^{d\times d}$ akkor és csak akkor DAG, ha

$$h(W) = tr(e^{W\circ W}) - d = 0. \tag{2.8}$$

▪

Bizonyítás. Az előző tételt terjesztjük ki olyan W gráfokra, melyekre $W_{ij} \in \mathbb{R}$. Valójában már az előző bizonyítás során sem használtuk fel azt, hogy $B_{ij} \in \{0, 1\}$, csupán azt, hogy $B_{ij} = 0$, illetve $B_{ij} > 0$. Tehát az 1. tétel nemcsak a bináris, 0 vagy 1 élsúlyokat tartalmazó gráfokra igaz, hanem minden pozitív és 0 élsúlyú éleket tartalmazó gráfra.

Hogy a negatív élsúlyokat tartalmazó gráfokra is kiterjeszthető legyen a tétel, $tr(e^W)$ helyett vizsgáljuk $tr(e^{W \circ W})$-t, ahol $W \circ W$ a W mátrix önmagával vett Hadamard-szorzata: $[W \circ W]_{ij} = W_{ij}^2$. B mátrix helyett $W \circ W$-vel számolni azzal egyenértékű, mintha az irányított utak összeszámolásában minden élet W_{ij}^2 súllyal számolnánk. Vagyis ha $h(W) > h(W')$, az két dolgot jelenthet: vagy W-ben több irányított kör található, vagy nagyobb súlyuk van az ezekben szereplő éleknek. Ezzel megmarad a függvénynek két fontos tulajdonsága is: Egyrészt, az továbbra is fennáll, hogy $h(W) = 0$ akkor és csak akkor, ha W irányított körmentes. Másrészt, $h(W)$ továbbra is jellemzi, hogy W gráf mennyire áll messze a 'DAG-ságtól'.

Megjegyzés 2 *h függvény gradiense is könnyen számítható, mégpedig:*

$$\nabla h(W) = (e^{W \circ W})^T \circ 2W \tag{2.9}$$

Megjegyzés 3 *h függény és ∇h kiszámítása a mátrix exponens kiszámítását igényli, melyre létezik $O(d^3)$ algoritmus [1].*

2.2.3. A kiterjesztett Lagrange-típusú formula

A 2.4-es probléma megoldását a kiterjesztett Lagrange-típusú formula alkalmazásával javasolja a [16] tanulmány. Ezzel az optimalizációs módszerrel a kényszert tartalmazó probléma átvihető egy kényszer nélküli probléma megoldásába. Ez a Lagrange-féle multiplikátort használó módszeren alapul, azonban a két módszer nem ugyanaz. A probléma kiterjesztett Lagrange-típusú formulája:

$$L^\rho(W, \alpha) = F(W) + \frac{\rho}{2}|h(W)|^2 + \alpha h(W) \tag{2.10}$$

A kiterjesztett Lagrange-típusú módszer abban különbözik a Lagngrange-féle multiplikátort használó módszertől, hogy tartalmazza a $\frac{\rho}{2}|h(W)|^2$ tagot is. Lényegében a kiterjesztett Lagrange-típusú formula ekvivalens az alábbi probléma Lagrange-féle multiplikátort használó módszer Lagrange-függvényével [5]:

$$\begin{aligned} &\min_W F(W) + \frac{\rho}{2}|h(W)|^2 \\ &\text{feltéve, hogy } h(W) = 0 \end{aligned} \tag{2.11}$$

A kiterjesztett Lagrange-típusú módszert használva a megoldási algoritmus egy duális optimalizálási feladat:

$$D(\alpha) = \min_{W \in \mathbb{R}^{d \times d}} L^\rho(W, \alpha) \tag{2.12}$$

A cél a duális optimalizálási feladat lokális megoldását megtalálni a 2.13 szerint:

$$\max_{\alpha \in \mathbb{R}} D(\alpha) \tag{2.13}$$

A 2.10-es egyenlet alapján $D(\alpha)$ α szerinti gradiense könnyen kiszámítható: $\nabla D(\alpha) = h(W_\alpha^*)$, ahol W_α^* a 2.10 lokális minimuma. Ebből adódik α paraméter frissítése:

$$\alpha \leftarrow \alpha + \rho W_\alpha^* \tag{2.14}$$

A kiterjesztett Lagrange-típusú formula segítségével tehát visszavezettük a kényszert tartalmazó problémát egy kényszer nélküli probléma megoldására. Ezzel azonban még nem vagyunk készen, hiszen a 2.12-es problémát is meg kell oldani. A tanulmány erre a közelítő kvázi Newton-módszert javasolja [17].

2.2.4. Az algoritmus működése

A NOTEARS algoritmus ae 1. pszeudokóddal foglalható össze [16].

1. Algoritmus NOTEARS

1. Paraméterek inicializálása: (W_0, α_0), max_iter megállási küszöb, $c \in (0, 1)$ lépésköz, h_{tol} küszöbérték, λ regulariációs paraméter, ω vágási küszöb.
2. t= 0, 1, 2, ... max_iter
   (a) $W_{t+1} \leftarrow argmin_W L^\rho(W, \alpha_t)$ olyan ρ-val, hogy $h(W_{t+1}) < ch(W_t)$
   (b) $\alpha_{t+1} \leftarrow \alpha_t + \rho h(W_{t+1})$
   (c) Ha $h(W_{t+1}) < h_{tol}$ return $W_{sol} = W_{t+1}$
3. Vágási küszöb alkalmazása W_{sol} gráfon: $\widehat{W} := W_{sol} \circ 1(|W_{sol}| > \omega)$

2.3. A NOTEARS-ADMM algoritmus

Az élet számos területén találkozhatunk elosztott adatokkal, melyek érzékenyek is lehetnek. Federált tanulási módszereket alkalmazva lenne lehetőség DAG-struktúra tanulására anélkül, hogy az adatokat meg kellene osztani egymással vagy egy központi szerverrel. A 2.2-es alfejezetben bemutatott NOTEARS algortimus azonban ebben a formájában nem alkalmas elosztott tanulásra. Ennek oka az, hogy a 2.4-es probléma esetén $F(W)$ hiába szeparálható, a 2.10-es kiterjesztett Lagrange-típusú formula már nem az [5]. Erre nyújt megoldást a *Towards Federated Bayesian Network Structure Learning with Continuous Optimization* [12] című tanulmányban megjelent *Distributed BNSL with ADMM* algoritmus, melyet a továbbiakban a szerzőkhöz hasonlóan NOTEARS-ADMM-nek rövidítek.

Tekintsük az alábbi problémát [12]: Keressük azt a $G = (V, E)$ gráfot, amely az $X = (X_1, X_2, ...X_d)$ valószínűségi vektorváltozó együttes $p(X_1, X_2, ...X_d)$ eloszlását reprezentálja. K darab partner lokális adathalmazai egyenként n_k darab mintát tartalmaznak $p(X)$ együttes eloszlásból. $n = \sum_{k=1}^{K} n_k$ a teljes adathalmaz mintszáma, mely az egyes partnerek lokális adathalmazai mintaszámainak összegeként adódik. A cél DAG olyan módon való tanulása adatból, hogy a partnerek x_k lokális adathalmazát ne kelljen megosztani.

Az ADMM algoritmus a duális optimalizálás szeparálható tulajdonságát és a kiterjesztett Lagrange-típusú formula konvergencia tulajdonságát ötvözi [5]. Így a kiterjesztett Lagrange-típusú formula helyett az ADMM (*Alternating Direction Method of Multipliers*) módszert használva már van lehetőség a 2.4-es probléma dekompozíciójára:

$$\min_{B_1, B_2, ...B_k, W} \sum_{k=1}^{K} \ell(B_k; \mathbf{x}_k) + \lambda \|W\|_1 \tag{2.15}$$

ahol $B_1, B_2, ...B_k \in \mathbb{R}^{d \times d}$ rendre a K darab partner lokális változói, W pedig a közös globális változó. A 2.10-eshez hasonlóan itt is felírható a kiterjesztett Lagrange-típusú formula annak figyelembevételével, hogy a korábbi $F(W)$ függvényt a felbontottuk 2.15

szerint:

$$L(B_1, ...B_k, W, \alpha, \beta_1, ...\beta_k; \rho_1, \rho_2) = \sum_{k=1}^{K} \ell(B_k; \mathbf{x_k}) + \lambda\|W\|_1 + \alpha h(W) + \frac{\rho_1}{2} h(W)^2 + \\ + \sum_{k=1}^{K} tr(\beta_k (B_k - W)^T) + \frac{\rho_2}{2} \sum_{k=1}^{K} \|B_k - W\|_F^2 \tag{2.16}$$

Az ADMM módszert alkalmazva pedig a következőképpen alakulnak az egyes paraméterek értékeinek frissítései (ahol $S_k = \frac{1}{n} \sum_{i=1}^{n_k} x_{k,i} x_{k,i}^T$):

$$B_k^{t+1} := (S_k + \rho_2^t I)^{-1} (\rho_2^t W^t - \beta_k^t + S_k) \tag{2.17}$$

$$W^{t+1} := argmin_W \Big(\lambda\|W\|_1 + \alpha^t h(W) + \frac{\rho_1^t}{2} h(W)^2 + \\ + \sum_{k=1}^{K} tr(\beta_k^t (B_k^{t+1} - W)^T) + \frac{\rho_2^t}{2} \sum_{k=1}^{K} \|B_k^{t+1} - W\|_F^2 \Big) \tag{2.18}$$

$$\alpha^{t+1} := \alpha^t + \rho_1^t h(W^{t+1}) \tag{2.19}$$

$$\beta_k^{t+1} := \beta_k^t + \rho_2^t (B_k^{t+1} - W^{t+1}) \tag{2.20}$$

$$\rho_1^{t+1} := \rho_1^t \gamma_1 \tag{2.21}$$

$$\rho_2^{t+1} := \rho_2^t \gamma_2 \tag{2.22}$$

A 2.17-2.22 egyenletek alapján adódnak a NOTEARS-ADMM algoritmus lokális lépései, melyek az egyes partnereknél számítódnak ki párhuzamosan, illetve a globális, központi lépések, melyeket a központi szerver hajt végre. A terjedelem miatt nem részleteztem az egyes lépések kiszámítási módja mögötti elméletet, az eredeti tanulmányban megtalálhatók a lépések részletes leírásai [12], itt [5] pedig az ADMM algoritmus elméleti háttere.

2. Algoritmus NOTEARS-ADMM

1. Paraméterek inicializálása: $\rho_1, \rho_2, \alpha^1, \beta_1^1, ..., \beta_K^1$, $\gamma_1, \gamma_2 > 1$, W^1, max_iter megállási küszöb, λ regularizációs paraméter, ω vágási küszöb.
2. for t= 0, 1, 2, ... max_iter do
   (a) Minden partner megoldja a 2.17-es egyenletet párhuzamosan, lokálisan.
   (b) A partnerek elküldik a központi szervernek $B_1^{t+1}, B_2^{t+1}, ...B_K^{t+1}$ modellparamétereket.
   (c) A központi szerver megoldja a 2.18-as egyenletet.
   (d) A központi szerver elküldi W^{t+1}-et minden partnernek.
   (e) A központi szerver frissíti a paramétereket a 2.19, 2.20, 2.21, és 2.22-es egyenletek szerint.
   (f) Minden partner frissíti a lokális paramétereit a 2.20 és 2.22-es egynletek szerint

2.4. A NIC adatbázis

A neonatális intenzív centrumok koraszülöttek intenzív ellátását végzik. Az országban 22 ilyen regionális centrum található, a NIC adatbázis adatai ezekből a centrumokból származnak. 2002 óta valósul meg az adatgyűjtés a centrumokban. Én a 2005 és 2013 között rögzített adatokkal dolgoztam, ebben az időszakban összesen közel *56 ezer* koraszülött adatait rögzítették, összesen *338* változó szerint. Ezek közül csak körülbelül *180* darab változó kitöltése releváns az esetek nagy részében.

Amikor egy koraszülött bekerül a rendszerbe, rögzítik annak a neonatális intenzív centrumnak a kódját, ahol a kezelése megkezdődik. Időnként előfordul, hogy egy koraszülöttet át kell szállítani másik intézménybe, például valamilyen speciális ellátás szükségessége miatt, ami csak egy nagyobb centrumban biztosítható. Ekkor rögzítésre kerül a második, illetve ha több szállítás is történt, minden további intézmény azonosítója is. (Fontos azonban megjegyezni, hogy ez a fajta szállítás nem ugyanaz, mint amikor a koraszülöttet a születés helyéről azért kell másik intézménybe szállítani, mert csak ott működik neonatális intenzív centrum. Ekkor nyilvánvalóan annak a centrumnak a kódja kerül rögzítésre az adatbázisban, ahol a koraszülött intenzív ellátását megkezdték.)

A szállítás kérdésével mindenképpen érdemes foglalkozni, hiszen többféle lehetőség is kínálkozik a centrumok között szállított koraszülöttek adatainak kezelésére: Egyrészt ezeket figyelembe lehetne venni minden egyes centrumnál, akiknél kezelték a koraszülöttet. Ez viszont azt is jelenti, hogy bizonyos adatok nagyobb, kétszeres, háromszoros súllyal kerülnek be az elemzésbe, ami a legtöbb esetben nem kívánt mellékhatás.

Másrészt, ha csak egyszer vesszük figyelembe ezeket az adatokat, el kell dönteni, az első felvevő intézményhez, vagy valamelyik későbbi intézményhez legyenek-e sorolva. Ez azonban alapvetően egy orvosi döntés kell, hogy legyen, hiszen a jó választás attól függ, hogy mit szeretnénk megfigyelni az adatelemzés során: az adott cetrumokhoz forduló, ott a rendszerbe bekerülő koraszülöttekről szeretnénk információt nyerni, vagy arról, hogy az egyes centrumokból távozó koraszülöttek adatai között milyen összefüggések vannak.

Mivel a dolgozatom alapvetően egy mérnöki dolgozat, nem foglalkoztam behatóan a szállítás kérdésével: minden koraszülött adatát annál az intézménynél vettem figyelembe, ahol először rögzítették. Azonban a további munka és elemzések során mindenképpen érdekes lehet ezzel a kérdéssel foglalkozni.

A NIC nagy számú változója közül orvosszakértővel egyeztetve választottuk ki a változók egy olyan halmazát, amely a fontos és numerikus változókat tartalmazza. Mivel 27 változó még így is túl sok ahhoz, hogy egyszerre be lehessen fogadni és értelmezni a közöttük megjelenő kapcsolatokat, ezt tovább szűkítettem 9 olyan változóra, ami a koraszülöttek tüdőbetegségére és légzéstámogatására fókuszál.

Az alábbiakban ezt a 9 változót mutatom be, röviden összefoglalva a jelentésüket:

- **Gesztációs kor**: a koraszülött érettségét jellemző paraméter, a fogamzástól számítva a megszületésig eltelt hetek száma.
- **Születési súly**: a koraszülött születéskori súlya grammban mérve.
- **Apgar 1 perces**: egy 0 és 10 közötti érték, amely a koraszülött általános állapotát jellemzi születés után 1 perccel.
- **Antibiotikum kezelés időtartama**: a koraszülött antibiotikum kezelésének az időtartama napban mérve.
- **Surfactant adás alkalmak száma**: gyakran szükséges surfactant készítményt adni a koraszülötteknek, amely a tüdejük megfelelő működését segíti elő. Az ilyen kezelések számát rögzíti ez a változó.

- **Lélegeztetés összes napjainak száma**: hány napig részesült valamilyen fajta lélegeztetésben a koraszülött.
- **Nasalis CPAP időtartama**: CPAP készülékkel történő lélegeztetés időtartama napban mérve.
- **Konvencionális respiráció időtartama:** hagyományos módon történő lélegeztetés időtartama.
- **Oxigénadás időtartama**: oxigén alkalmazásának időtartama a légzés segítésére, napokban mérve. A *Nasalis CPAP* és a *Konvencionális respiráció napjait* egyszerre oxigénadásnak is minősítik, ezért ez a mérőszám lehet nagyobb, mint a *Lélegeztetés összes napjainak* száma.

Megjegyzés 4 *A Lélegeztetés összes napjainak a száma egy számított adat: a Nasalis CPAP időtartama, a Konvencionális respiráció időtartama és egy harmadik változó (Magas frekvenciájú oszcillációs lélegeztetés (HFOV) időtartama) összegeként adódik. Az utóbbi, harmadik változót nem vizsgáltam. Ennek oka egyrészt az orvosszakértő javaslata volt, aki nem jelölte magas prioritásúnak, másrészt ez a kapcsolat egy jó lehetőség az algoritmus tesztelésére: meg kellett találnia a számított változó és a másik kettő összefüggését (ami, mivel az összeg harmadik tagja hiányzik, valamivel nehezebb feladat).*

A tesztek során előkerül még az *Előző várandósságok száma* változó, mely a koraszülött édesanyja előző várandósságainak számát jelenti. Ezt a változót azonban a vizsgált tesztesetekben egyetlen másik változóhoz sem kötötte hozzá az algoritmus, így ennek a megjelenítését mellőztem. A kipótlási tesztek, illetve más, teljes változóhalmazt érintő tesztek során azonban megjelenik a dolgozatban.

2.5. Egyéb felhasznált definíciók, tételek

A dolgozatom 3. és 4. fejezetében használok még néhány további matematikai fogalmat, melyekre az algoritmusok által generált gráfok elemzése végett van szükség. Az alábbiakban ezeket a fogalmakat mutatom be, adok rájuk definíciót.

Két gráf különbözőségének számszerűsítésére alkalmasak a Levenshtein-távolságok (a későbbiekben LT-ként rövidítve, angol nyelvű szakirodalomban *graph edit distance*) [6]:

Definíció 5 (Irányított Levenshtein-távolság). Legyenek G_1 és G_2 irányított éleket tartalmazó gráfok. Ekkor G_1 irányított Levenshtein-távolsága G_2-től az a legkisebb szám, ahány

- él hozzávételével,
- él elvételével vagy
- él megfordításával

G_1 izomorffá alakítható G_2-vel. ▪

Definíció 6 (Irányítatlan Levenshtein-távolság). Legyenek G_1 és G_2 irányítatlan éleket tartalmazó gráfok. Ekkor G_1 irányítatlan Levenshtein-távolsága G_2-től az a legkisebb szám, ahány

- él hozzávételével
- vagy él elvételével

G_1 izomorffá alakítható G_2-vel. ▪

Egy másik probléma, ami a dolgozatomban előkerül, hogy hogyan lehetne több gráfból egyetlen *konszenzus-gráfot* kiszámítani, ami valamilyen módon a gráfok középértékét szemlélteti. Az a $G_1, G_2, ...G_N$ gráfokból számítható *konszenzus-gráf* definíciója a következő:

Definíció 7 (konszenzus-gráf). Legyenek $G_1, G_2, ...G_N \in \mathbb{D}$ gráfok, ahol $\mathbb{D}$ az irányított körmentes gráfok tere. Tekintsük M mátrixot, mely az alábbi módon képezhető $G_1, G_2, ...G_N$-ből: $M_{ij} = x$ akkor és csak akkor, ha $P(X) = x$, ahol X azt az eseményt jelöli, hogy a $G_1, G_2, ...G_N$ gráfok közül egy G_i gráfot véletlenszerűen kiválasztva lesz él G_i i-edik és j-edik csúcsa között.

M^* mátrixot M mátrixból képezzük olyan módon, hogy $M^*_{ij} = 1$ akkor és csak akkor, ha $M_{ij} > 0.5$, különben $M^*_{ij} = 0$. Ekkor nevezzük M^* által reprezentált gráfot a $G_1, G_2, ...G_N$ gráfok *konszenzus-gráfjának.* ▪

Megjegyzés 5 *M^* által reprezentált gráf alatt egy olyan A gráfot értünk, melyre igaz, hogy pontosan akkor van az i-edik csúcsából a j-edik csúcsba mutató él, ha $M^*_{ij} = 1$.*

3. fejezet

A kiterjesztett elemzési keretrendszer

Ebben a fejezetben bemutatom az elkészült keretrendszer részeit és ezek felelősségeit: röviden bemutatom az implementációt, majd áttérek a hiányos adatok kipótlásának kérdéskörére. Ezután részletesen bemutatom a NOTEARS és a NOTEARS-ADMM algoritmusok hiperparaméter-beállításának lépéseit.

A dolgozatom következő fejezeteiben közlök olyan eredményeket, melyek az egyes NIC centrumok lokális adatain futtatott algoritmus eredményei. Az adatok intézményi szintű védettségére tekintettel az intézmény kódja helyett egy véletlenszerűen generált betűkóddal jelölöm az egyes centrumokat. Az egyes centrumok konkrét mintaszámának közlését is szándékosan kerülöm, a centrum adathalmazának méretére csak az alábbi három kategória egyikébe való besorolással utalok: 'nagy', 'közepes', 'kicsi'.

3.1. Az implementáció

Jelen munka keretében egy a NIC adatbázisra szabott keretrendszert valósítottam meg Python nyelven, mely képes a 2.2 fejezetben bemutatott NOTEARS és a 2.3 fejezetben bemutatott NOTEARS-ADMM algoritmusok futtatására az adaton, a futási eredmények összehasonlítására, különböző paraméterezésű futtatások összemérésére. Képes továbbá a hiányos adatok kipótlására eloszlás alapján, valamint a kipótlás robusztusságának mérésére.

A keretrendszert Python nyelven valósítottam meg, melyhez felhasználtam a NOTEARS algoritmust bemutató tanulmány [16] által közzétett Python implementációt (NOTEARS implementáció elérhető itt), valamint a [12] tanulmány melllé közzétett algoritmus implementációt is (NOTEARS-ADMM implementáció elérhető itt).

A futtatásokat rövidebb tesztek esetén a lokálisan a saját számítógépemen végeztem (AMD Ryzen 7 4700U 8 magos processzor), más esetben a BME Méréstechnika és Információs Rendszerek Tanszékének TITAN nevű számítógépén (Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz, 56 mag). Összességében elmondható, hogy a megfelelő paraméterek mellett a NOTEARS-ADMM futása gyorsabb volt, mint a NOTEARS algoritmusé. Ez valószínűleg azért lehet, mert az ADMM optimalizációs módszer tulajdonsága, hogy viszonylag gyorsan talál egy optimumhoz közeli megoldást, nagy pontossággal azonban csak nagyon lassan közelít [5]. Az is megállapítható volt azonban, hogy a NOTEARS algoritmus futási ideje aránylag kis mértékben változott különböző paraméterbeállítások mellett, a NOTEARS-ADMM viszont nem megfelelő beállítások esetén nagyságrendekkel tovább futott. Ezen felül a futásidők mindkét algoritmus esetén jelentősen növekedtek a változószám (és a mintaszám) növelésével. A legegyszerűbb futási eseteket összehasonlítva elmondható,

hogy a lokális futtatás során, 9 változóval globális adaton (ez körülbelül 56 ezer egyéni adatot jelent) mindkét algoritmus 2 perc alatt befejezte a futását, a NOTEARS-ADMM valamivel hamarabb.

A dolgozatban bemutatott grafikonok egy részét szintén a Python kóddal generáltam, ehhez a Matplotlib könyvtárat használtam. (A többi ábra TikZ-ben készült.)

3.2. A hiányzó adatok pótlása

A NIC adatbázis elemzése során az első probléma, amivel szemben találtam magam, a hiányzó adatok kérdése volt. Az orvosszakértő által fontosnak megjelölt változók kitöltöttsége is meglehetősen tág keretek között változott: 99.96%-tól (ez a *Gesztációs kor* változó) 85.72%-ig (ez az *Oxigén adás időtartama*). Erre a problémára még a tényleges adatelemzés előtt megoldást kellett találnom, mivel a választott DAG tanulási módszer teljes adatot igényelt.

A hiányzás kezelésére többféle megközelítés szóba jöhet: a legegyszerűbb megoldás a hiányzó adatok eldobása. Ezzel azonban több probléma is van: Egyrész nem feltételezhető, hogy a hiányzás véletlenszerű. Sőt, az inkább valószínűsíthető, hogy bizonyos körülmények együttállása eredményezi egy változó hiányosságát: például egy adott centrumban valamit következetesen máshogy rögzítenek, vagy egy másik adatmező kitöltöttsége miatt nem rögzítik külön a változót. Hogyha ilyen esetben eldobnánk a hiányos adatokat, nagy valószínűséggel azokat az adatokat dobnánk el specifikusan, ahol bizonyos körülmények együttállása megvalósul.

Másrészt, a hiányzási százalékok nagyon magasak a NIC esetében, ha minden olyan adatrekord eldobásra kerülne, amelyiknek van kitöltetlen mezője, csak kevés adat maradna. Összességében tehát a hiányos adatrekordok eldobása nagyon szakszerűtlen módszer lenne a NIC adatbázis esetében.

Egy másik módszer a legközelebbi szomszédok módszere, mely esetén egy hiányzó érték esetén megkeressük valamilyen definiált távolság szerint az adatrekordhoz legközelebb eső másik, nem hiányos adatrekodot, és annak az adatával pótlunk. Ez a módszer akkor alkalmazható jól, ha az adatrekordok jól elválasztható csoportjai hasonlóan viselkednek. A NIC esetében ez sem áll fenn.

Így egy harmadik módszert, a véletlen kipótlást alkalmaztam. Ennek során egy változó eloszlása alapján random értékekkel történik a hiányzó adatok kipótlása. Ez a NIC esetében azért is jól alkalmazható, mert a változók nagy része feltételezhetően normális eloszlású (vagy ahhoz közel esik).

Az adatok védettsége miatt azonban ideális esetben nem szeretnénk, hogy az egyes centrumoknak meg kelljen osztani egymással az adataikat amiatt, hogy a teljes eloszlás alapján lehessen pótolni. (Lehetséges lenne centrumonként a lokális adatok alapján megbecsülni az eloszlást és ezáltal pótolni, azonban a globális adatot használva jóval pontosabb becsléseket kaphatunk.) Ahhoz, hogy egyértelműen azonosítható legyen a közös normális eloszlás, szükség van a globális átlagra és szórásra. Ezeket szerencsére ki lehet számolni a partnerek egyéni átlagai és szórásai, mint elégséges statisztikák alapján, az adatok megosztása nélkül.

A közös átlag (x_c) képlete k darab partner esetén, ahol a partnerek egyenkénti mintaszámai $n_1, n_2, ...n_K$, az átlagai pedig $\overline{x_1}, \overline{x_2}, ...\overline{x_K}$:

$$\overline{x_c} = \frac{n_1\overline{x_1} + n_2\overline{x_2} + ... + n_K\overline{x_K}}{n_1 + n_2 + ... + n_K} \tag{3.1}$$

A közös szórás kiszámítására alkalmas képlet már nem ennyire magától értetődő, [11] alapján vezettem le a közös szórás képletét K partner esetére: S_c a közös szórás, $s_1, s_2, ...s_K$ rendre a K darab partner egyedi szórásai.

Egy j-edik partner szórása definíció alapján így számolható (ahol n_j a partner mintaszáma, $x_j i$ egy adat, $\overline{x_j}$ pedig az $x_j i$ adatok átlaga:

$$S = \sqrt{\frac{1}{n_j - 1} \sum_{i=1}^{n_j} (x_{ji} - \overline{x_j})^2} \tag{3.2}$$

A 3.2-es definíció alapján a teljes adathalmaz szórásnégyzete:

$$S_c^2 = \frac{1}{n_1 + n_2 + ...n_K - 1} \left(\sum_{i=1}^{n_1} (x_{1i} - \overline{x_c})^2 + \sum_{i=1}^{n_2} (x_{2i} - \overline{x_c})^2 + ... + \sum_{i=1}^{n_K} (x_{Ki} - \overline{x_c})^2 \right) \tag{3.3}$$

Vegyük észre, hogy:

$$\begin{aligned} (x_{ji} - \overline{x_c})^2 = (x_{ji} - \overline{x_j} + \overline{x_j} - \overline{x_c})^2 = \\ = (x_{ji} - \overline{x_j})^2 + 2(x_{ji} - \overline{x_j})(\overline{x_j} - \overline{x_c}) + (\overline{x_j} - \overline{x_c})^2 \end{aligned} \tag{3.4}$$

A 3.4-es pontban megfigyelt azonosság segítségével bontsuk ki a 3.3-mas egyenletben szereplő j-edik partner egyes adatai és az átlag különbségei négyzetének az összegét:

$$\sum_{i=1}^{n_j} (x_{ji} - \overline{x_c})^2 = (n-1)s_j^2 + 2(\overline{x_j} - \overline{x_c}) \sum_{i=1}^{n_j} (x_{ji} - \overline{x_j}) + +n(\overline{x_j} - \overline{x_c})^2 \tag{3.5}$$

A középső tag felbontható a következőképpen:

$$\sum_{i=1}^{n_j} (x_{ji} - \overline{x_j}) = \sum_{i=1}^{n_j} x_{ji} - \sum_{i=1}^{n_j} \overline{x_j} = x_{j1} + x_{j2} + ... + x_{jn_j} - n\overline{x_j} = 0 \tag{3.6}$$

3.6 miatt a középső tag kiesik. Ennek köszönhetően 3.3 és 3.5 alapján már felírható a közös szórás kiszámításához szükséges képlet az x_i értékek, azaz az egyes adatok ismerete nélkül:

$$S_c = \sqrt{\frac{\sum_{i=1}^{K} (n_i - 1)s_i^2 + n_i(\overline{x_i} - \overline{x_c})}{\left(\sum_{i=1}^{K} n_i \right) - 1}} \tag{3.7}$$

Ezt a képletet használva már tökéletesen egyezett az adatokból közvetlenül számított és a képlettel kiszámított szórás.

Természetesen a mintaszám, az átlag és a szórás megosztása is felvethet adatvédelmi kérdéseket, azonban az egyedi adatrekordok megosztása így elkerülhető. Egy további irány lehet annak vizsgálata, hogyan lehetne még kevesebb információ megosztásával hatékonyan kipótolni a hiányzó adatokat.

Egy következő előkészítő lépés volt a változók eloszlásának standardizálása, mely az eredeti X normális eloszlásból a 3.8-as képlettel számítható ki:

$$Z = \frac{X - \mu}{\sigma} \tag{3.8}$$

Így már $Z \sim N(0, 1)$ standard normális eloszlásokkal számolhattam. Mivel a 3.1-es és a 3.7-es képletekkel megkapható a teljes adat átlaga és szórása, lehet ez alapján standardizálni, ami szintén pontosabb eredményt ad, mintha a lokális adatok alapján tennénk ugyanezt.

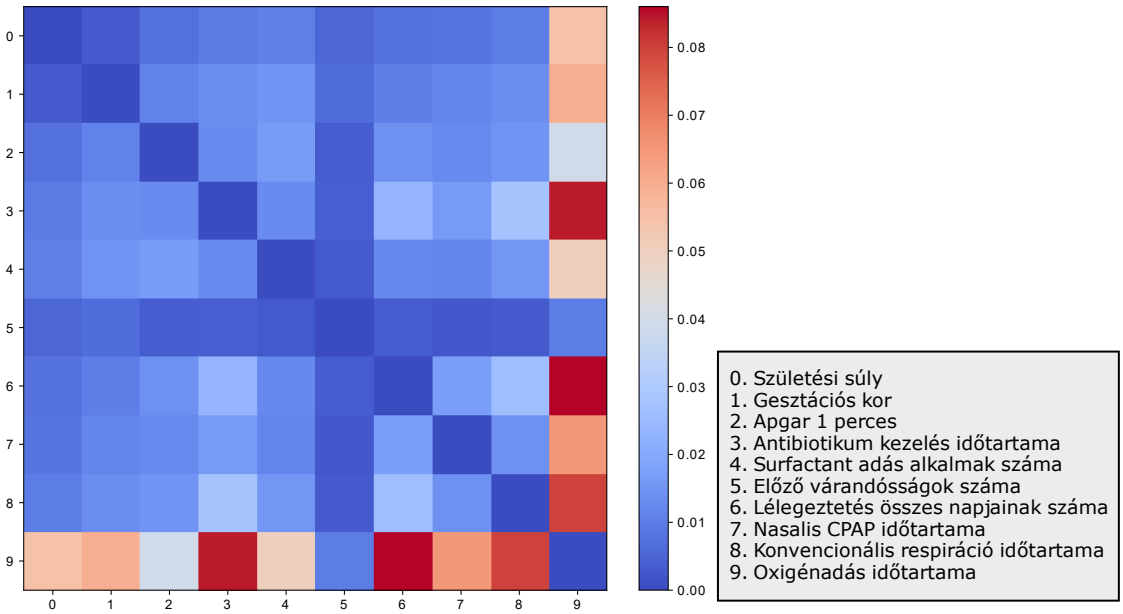
Ezek után a kipótlás robusztusságának tesztelése következett. A tesztek során az alábbiakat vizsgáltam:

1. A kipótlás előtti és kipótlás utáni korrelációs mátrixok különbségeit több különböző kipótlásra átlagolva. (A kipótlás előtti korrelációs mátrix számításánál figyelmen kívül hagytam az üres értékeket.) Ugyan egy standardizált normális eloszlás kovarianciamátrixa és korrelációs mátrixa megegyezik, mégis volt jelentősége annak, hogy korrelációs mátrixot számoltam. Például egyes centrumok kipótlásra való robusztusságát vizsgálva lehet, hogy a teljes eloszlásra nézve egy $\mu = 0$, $\sigma = 1$ eloszlást követnek rendre az egyes változók, de csupán a partner adatait nézve a lokális $\overline{x_i}$ átlagot és s_i szórást nézve ezek nem pont a 0 és 1 értékeket vették fel. Így apróbb torzítások kerülhettek volna be az összehasonlító elemzésbe, ennek kiküszöbölése végett számoltam korrelációs mátrixszal.

2. Vizsgáltam különböző kipótlások után ugyanazzal a paraméterezéssel generált gráfok különbségeit (irányított és irányítatlan élként is értelmezve az éleket). A paraméterek beállításáról részletesen a 3.3 fejezetben írok, itt csak megelőlegezem a $\lambda = 0.1$ és $\omega = 0.2$ beállításokat a NOTEARS algoritmus esetében.

Az alábbiakban egy szűkebb változóhalmazon mutatom be az adatkipótlás robusztusságát. A 3.1-es ábrán látható egy hőtérkép, amely a kipótlás előtti és utáni korrelációs mátrixok különbségeit szemlélteti. Minél sötétebb a mátrix egy M_{ij} cellájának színe, annál inkább elmondható, hogy az i-vel és j-vel jelölt változók között a kapcsolat nem változott a kipótlások során.

Először is fontos rögzíteni, hogy még a pirossal jelölt, legnagyobb eltérési értékek is 0.08 környékén mozognak, ami még mindig nagyon kicsinek mondható. Ugyanakkor egyértelműen látszódik, hogy az *Oxigén adás időtartama* változó korrelációja a többi változóval a legkevésbé stabil. Ez ugyanakkor nem meglepő, hiszen ennek a változónak a legalacsonyabb a kitöltöttségi szintje, mindössze 86%.



3.1. ábra. Az átlagos eltérés a kipótlás előtti korrelációs mátrixtól 10 kipótlás eredményei alapján

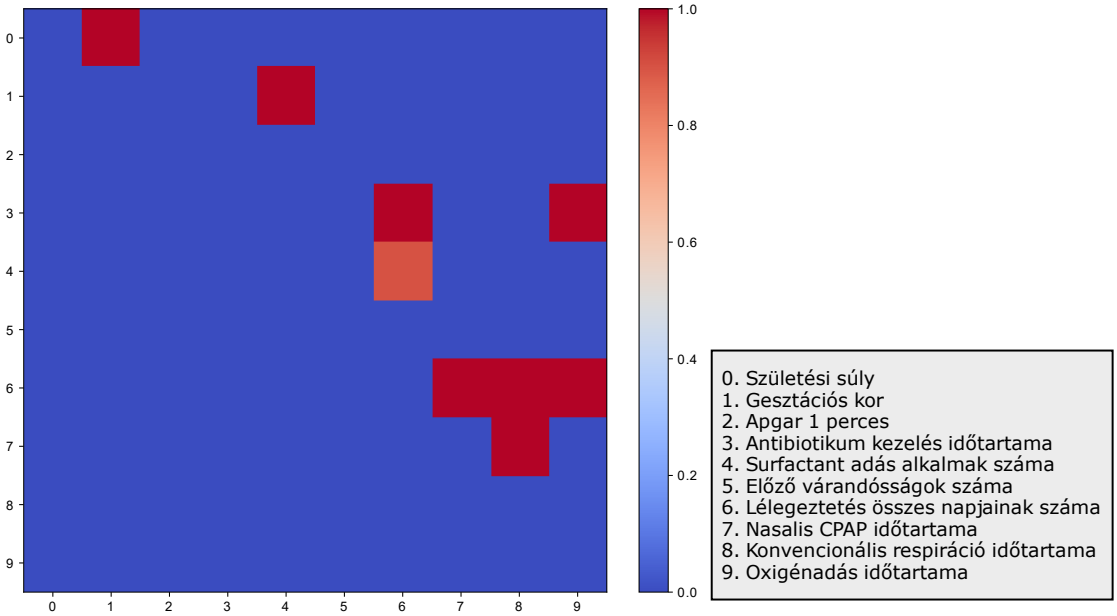
A 3.2-es és a 3.3-mas ábrán látható hőtérképek azt szemléltetik, hogy különböző kipótlások után az adathalmazon generált gráfok élei milyen valószínűséggel fordulnak elő: minden cella egy 0 és 1 közötti számot szemléltet: ha 0 szerepel a cellában, akkor a 10 futtatás 0%-ában jelent meg az él, ha 1, akkor 100%-ban jelen volt az él. Tehát lényegében a sok 0 közeli és a sok 1 közeli érték jelent jót. Az olyan élek, amelyek 0.5 közeliek azt mutatják, hogy a kipótlások nagyban befolyásolják az adatból generált struktúrát.

Az eredményekben viszont ilyen él nem szerepel: irányított és irányítatlan esetben is egyetlen él kivételével minden más élet vagy minden kipótlás után behúz az algoritmus, vagy egy futtatás után sem húz be. Az egyetlen él, ami nem jelenik meg minden futás során, 80%-ban így is jelen van, ami egy jó aránynak számít. Az is megfigyelhető, hogy az élek irányításában teljesen egységes az algoritmus: ha egy élet behúz, minden alkalommal ugyanabba az irányba húzza be.

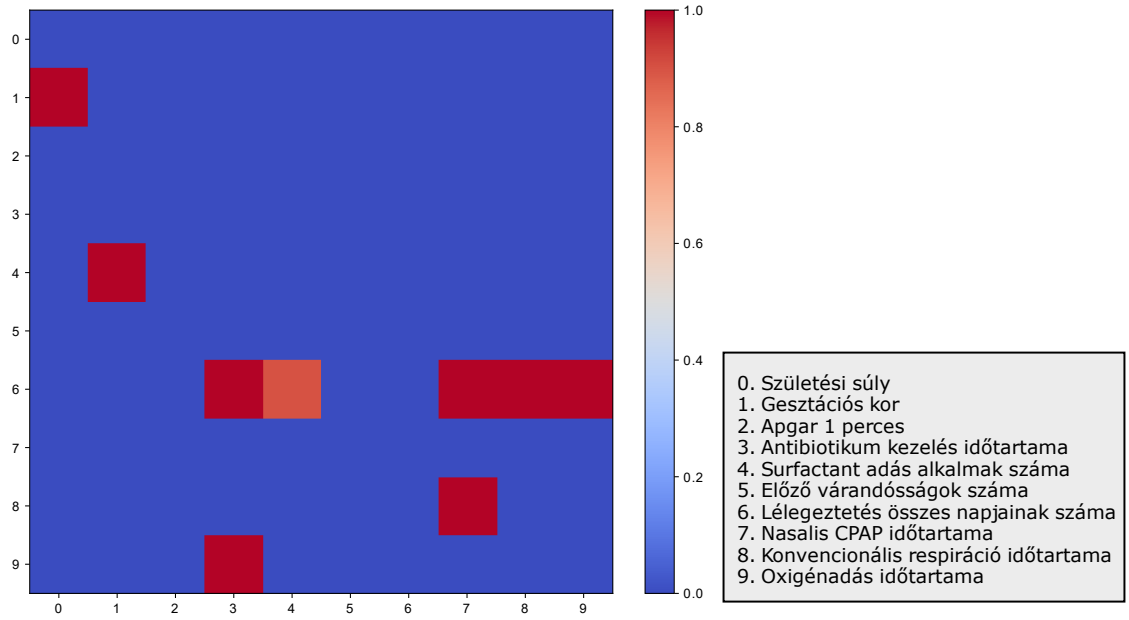
Az átlagos élszámra, és a gráfok egymástól mért Levenshtein-távolságára vonatkozó adatokat a 3.1-es táblázat tartalmazza összesítve:

Átlagos élszám	LT irányított	LT irányítatlan
8.9	0.2	0.2

3.1. táblázat. 10 kipótlás futtatása során generált gráfok különbségei.



3.2. ábra. 10 kipótlás során felrajzolt gráfok élei előfordulásának valószínűsége, az éleket irányítatlan élként kezelve



3.3. ábra. 10 kipótlás során felrajzolt gráfok élei előfordulásának valószínűsége, az éleket irányított élként kezelve

3.3. Hiperparaméter beállítások

A következő alfejezetben bemutatom, hogyan végeztem az algoritmusok paramétereinek a hangolását. Ezt részletesen egy alfejezetben tárgyalom, azonban többször szembesültem azzal a későbbi munka során is, hogy az addig megfelelőnek hitt paraméterbeállításokon még hangolnom kellett: így a paramétereket valójában iteratív módon hangoltam. Az alábbiakban a végső, legjobbnak talált beállításokat rögzítem.

3.3.1. NOTEARS algotitmus

Az algoritmusnak a következő paraméterei állíthatóak:

- λ paraméter: ℓ_1 reguarizáció paramétere
- h_{tol} küszöbérték: Tudjuk, hogy a generált W gráf akkor és csak akkor DAG, ha a $h(W) = 0$ feltétel teljesül. Gyakorlatban a számítógépes számábrázolás miatt azt vizsgáljuk, hogy $h(W) < h_{tol}$.
- loss_type: Az algoritmus által használt veszteség-függvény (loss-function), megmutatja, mennyire jól illeszkedik a generált gráf az adatokra. A munkám során végig a legkisebb négyzetek módszerét alkalmaztam, mivel az eredeti tanulmányban is ezt használták.
- max_iter: Biztosítja, hogy az algoritmus futása mindenképpen leálljon. Nem volt szükséges módosítani az értékét az alapértelmezett beállításhoz képest.
- ρ_{max}: A kiterjesztett Lagrange-típusú optimalizáció elég nagy ρ esetén a minimumhoz konvergál. A minimum megtalálásában ugyanakkor az is problémát jelent, ha ρ túl nagy. Ennek a kiküszöbölésére maximalizálni kell ρ értékét. Ezen az alapértelmezett beállításon sem kellett módosítanom, a $\rho_{max} = 10^{-16}$ megfelelő érték volt.

- c: lépésköz paraméter, segítségével $h(W)$ iterációnkénti csökkenésére vonatkozó feltétel adható meg
- α_0: a Lagrange-multiplikátor kezdeti értéke
- ω vágási küszöb: A generált gráf élei közül az algoritmus lefutása után szükséges eldobni azokat, amelyek nagyon gyenge kapcsolatot reprezentálnak két változó között. Ehhez egy egyszerű küszöbértéket használ az algoritmus.

A hiperparaméterek vizsgálatát a h_{tol} paraméter értékének beállításával kezdtem: a tanulmányban javasolt 10^{-8} érték mellett megvizsgáltam az algoritmus viselkedését 10^{-6} és 10^{-3} érték mellett is. A futásidő érdemben nem növekedett kisebb küszöbérték esetén sem. Ritkán az is megfigyelhető volt, hogy az algorimtus kevesebb élet húz be nagyobb h_{tol} érték mellett. (Ez érthető is, hiszen ha az algoritmus futása hamarabb megáll, a generált gráf egy kevésbé pontos közelítése lesz a valódi struktúrának.) Mivel matematikailag a kisebb érték a helyes, valamint mivel a futásidőt érdemben nem befolyásolta, a $h_{tol} = 10^{-8}$ értéket alkalmaztam a továbbiakban.

Ezután következett a λ paraméter, ℓ_1 reluarizáció paramétere és ω vágási küszöb beállítása. Ezeket nem lehetett szigorú sorrendben, egymás után végezni: egy-egy jónak tűnő ω paraméterhez igazított λ érték esetén újra meg kellett vizsgálni ω lehetséges értékeit.

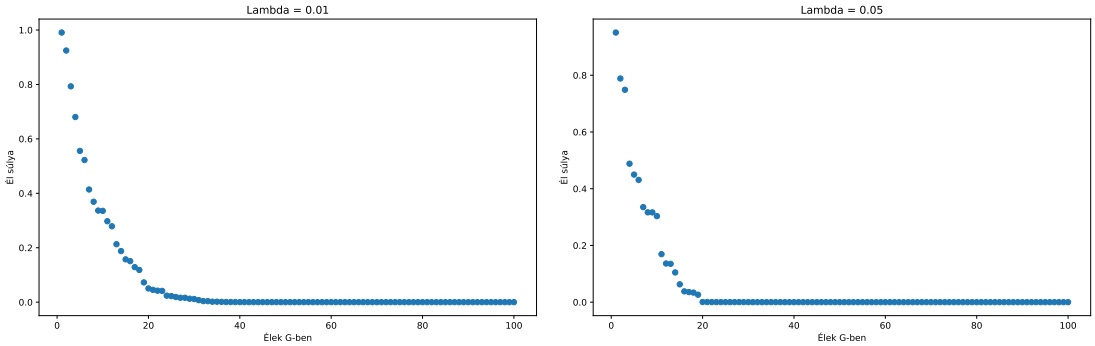
Az iterációt ω lehetséges értékeinek vizsgálatával kezdtem. Az algoritmust publikáló tanulmányban [16] az $\omega = 0.3$ szuboptimális értéket javasolják, így ez jó kiindulópont volt. A vágás célja tulajdonképpen az, hogy eltávolítsa azokat az éleket a gráfból, amelyek a véletlen zaj miatt keletkeztek. Ezen felül pedig szabályozható az is, hogy mennyire erős kapcsolatokat szeretnénk vizsgálni.

Az algoritmust először $\omega = 0$ paraméterrel futtattam, különböző λ értékek mellett. A 3.4-es és a 3.5-ös ábrákon ezeknek a futásoknak az eredményei láthatók: az x tengelyen minden beosztás egy-egy lehetséges irányított élnek felel meg, az y tengelyen pedig az élhez rendelt súly látható, mindezek csökkenő sorrendbe rendezve. (Nyilván a 0 súly azt jelenti, hogy az él nincs jelen.)

Megfigyelhető (a λ paraméter beállítására előreutalva), hogy λ értéke minél kisebb, az algoritmus annál több nem nulla súlyú élet húz be. Az is látszik, hogy a lehetséges élek nagyobb része a 0 értéket veszi fel. Megfigyelhető azonban leginkább a 3.5b, 3.5c részábrákon (de a 3.4b és 3.4c ábrákon is látszik) , hogy $\omega = 0.2, 0.3$ környékén van egy törés, onnantól felfele az élek súlyai közötti külöbségek még meredekebben nőnek. A többi ábráról is hasonló olvasható le, bár ez a határvonal nem mindenhol egyértelmű. Ezek alapján az $\omega = 0.2$ környéke egy jó paraméterválasztásnak tűnik: $\omega = 0.3$ csak a biztosabb kapcsolatokat hagyja meg, $\omega = 0.2$ viszont több kevésbé erős kapcsolatot is megőriz.

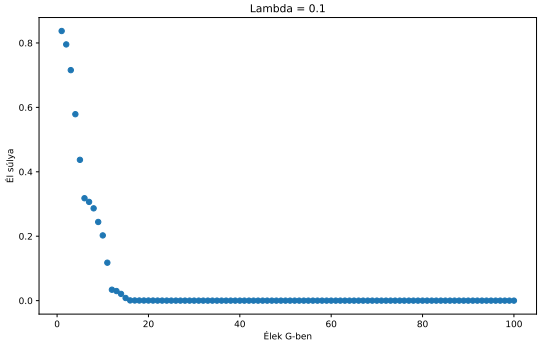
Itt szeretnék kitérni arra is, miért éppen néhány nagyobb mintaszámú partner adatát használtam ebben a fejezetben az algoritmus futtatásához: ugyanis lehetett volna a globális adatot, partnerek egy csoportjának az adatát, illetve kisebb partnerek adatait is használni. A paraméterek beállításánál az volt a cél, hogy olyan adatot használjak, ami (várhatóan) nagyobb mértékben robusztus a gráfépítésre. Azt tudhatjuk, hogy a partnerek struktúrái nem feltétlenül hasonlóak, sőt, előzetesen inkább az valószínűsíthető, hogy különbözni fognak. A globális adat éppen ezért nem biztos, hogy egy jó választás a megfelelő paraméterbeállítások megtalálására, ugyanis nem várt módon viselkedhet. (A globális struktúrát a későbbiekben részetesen vizsgálom.) Egy kis mintaszámú partner esetén szintén nehéz feltételezéseket tenni: lehet, hogy a partner a valamilyen szempontból épp a különleges eseteket kezeli, ami jelentős eltérést mutathat a többi partner struktúrájához képest.

Természetesen egy nagy mintaszámú partner esetén sem lehet kizárni az említett okokat (a futtatások alapján az is nyilvánvaló lesz, hogy a nagyobb partnerek adathalmaza

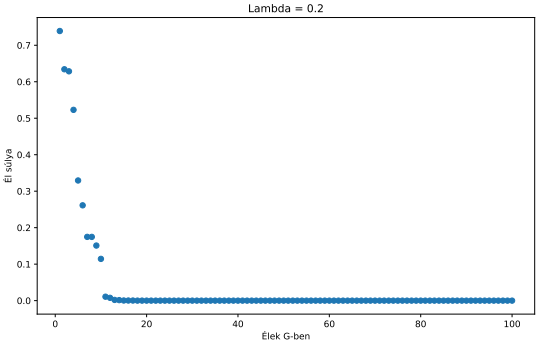


(a) $\lambda = 0.01$

(b) $\lambda = 0.05$

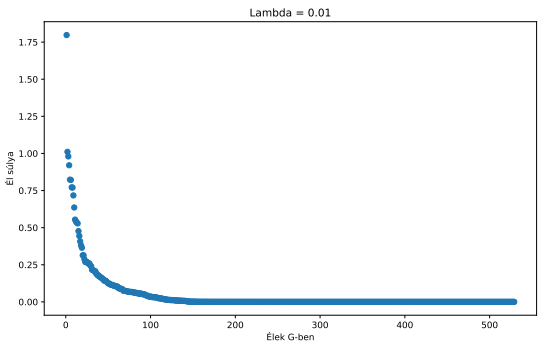


(c) $\lambda = 0.1$

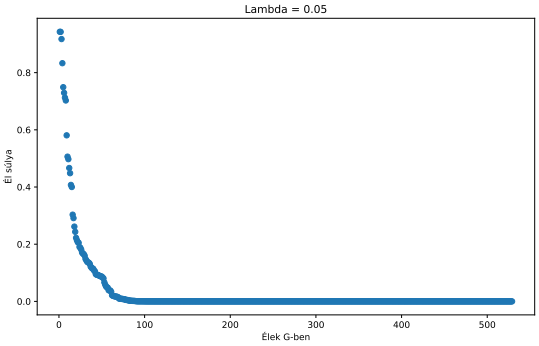


(d) $\lambda = 0.2$

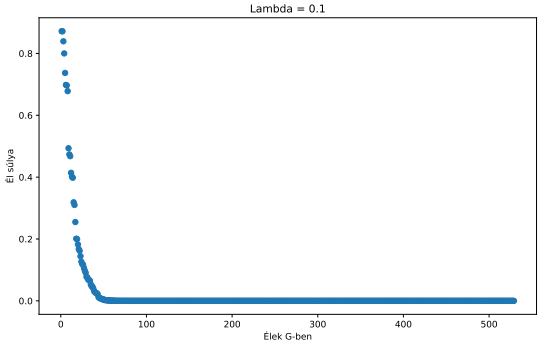
3.4. ábra. Az egyik nagyobb partner adataiból, d=10 változó esetén behúzott vágatlan élek súlyai



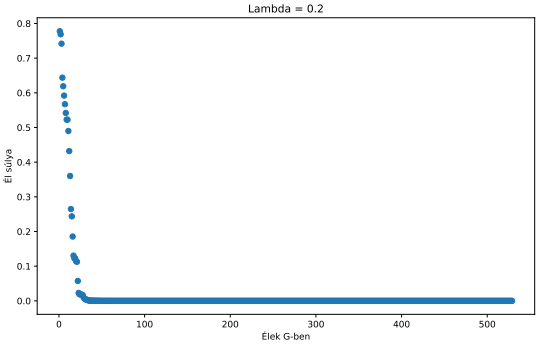
(a) $\lambda = 0.01$



(b) $\lambda = 0.05$



(c) $\lambda = 0.1$



(d) $\lambda = 0.2$

3.5. ábra. Az egyik nagyobb partner adataiból, d=22 változó esetén behúzott vágatlan élek súlyai

sem teljes mértékben robusztus). Mégis, a paraméterbeállítás során arra próbáltam törekedni, hogy a lehető legideálisabb körülményeket biztosítsam, és erre a legjobb választásnak a nagyobb adathalmazzal rendezlkező partnerek tűntek.

Olyan paraméterbeállítási teszteseteket is be fogok mutatni, amelyeknél nem a partner teljes adathalmazán (vagy a teljes globális adaton), hanem azoknak bootstrap [10] adathalmazain építettem gráfot. Gyakran használtam aránylag nagynak számító, n=500 és n=1000 közötti mintaszámú halmazokat. Ezt azért tettem, mert ezúton is biztosítani akartam az adat robusztusságát: ezeknek a teszteknek a lényege nem az, hogy megmutassák, az adat mennyire robusztus struktúraépítésra, hanem hogy egy minél robusztusabb adathalmazon vizsgálni lehessen, hogy melyik paraméterbeállítás hozza leginkább a biztos eredményt.

A λ paraméter beállíása volt a következő lépés: a 3.4-es és 3.5-ös ábrákon megfigyelhető az is, hogy λ értékének csökkenésével az algoritmus egyre több élet húz be. Ez nem meglepő, hiszen λ ℓ_1 regularizációs paraméter a 0 súlyok megjelenését ösztönzi, ritkább mátrixot adva eredményül nagyobb λ értéknél. Nem szabad azonban túl kicsire sem állítani λ-t, mert akkor túltanulás léphet fel: olyan összefüggésekre is rátanulhat az algoritmus, amelyek csak a mintahalmaz véges természete miatt állnak fenn. Ez például azon érhető tetten, hogy megjelennek a gráfban egészen kicsi (10^{-7}) értékek, melyek sokszor a gráf DAG-ságát is elrontják, és nyilvánvalóan jelentéktelenek az általános 10^{-1} nagyságrendű élekhez képest.

Ezeket szem előtt tartva, futási kísérletekkel tudtam meghatározni azt az intervallumot, amelyben λ mozoghat: azt mondhatjuk, $\lambda \in [0.01, 0.5]$. Ezen belül is $\lambda = 0.5$ esetén az algoritmus sokszor csak nagyon kevés élet húz be, míg $\lambda = 0.01$ esetén nem kizárt, hogy zaj jelenik meg a gráfban.

Ezeken a határokon belül azonban nehéz eldönteni, hogy mely λ érték a megfelelő. Abból lehet kiindulni, hogy egy paraméterbeállítás akkor jó, ha nagyjából hasonló gráfokat generál az adatból véletlenszerűen választott bootstrap halmazokon. Ha ezek a gráfok nagyon különbözőek, az rossz beállításra utalhat.

Gráfok különbségét kétféle metrikával is vizsgáltam, ezeknek formális definíciói megtalálhatók a 5. és 6. pontokban:

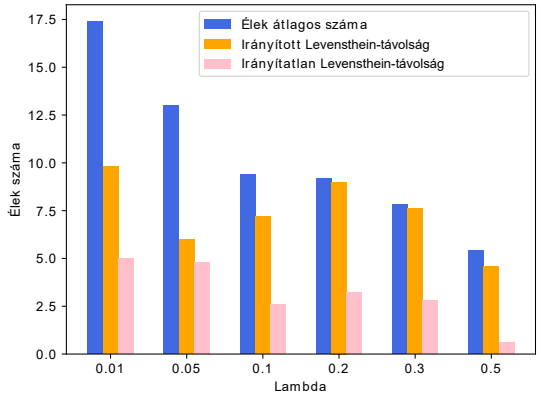
- Levenshtein-távolság irányított élek mentén
- Levenshtein-távolság irányítatlan élek mentén

Ugyanakkor ezek ismerete nem elégséges, hiszen az, hogy a Levenshtein-távolság két gráf között $LT = n$, még önmagában nem árulja el, mennyire találta meg a hasonló összefüggéseket az algoritmus. Azt is szükséges tudni, hogy összesen hány él van átlagosan a gráfokban, amik közül n különbözik. Tehát végsősoron az $\frac{LT}{e}$ hányados a mérvadó, ahol e az összes behúzott él számát reprezentálja.

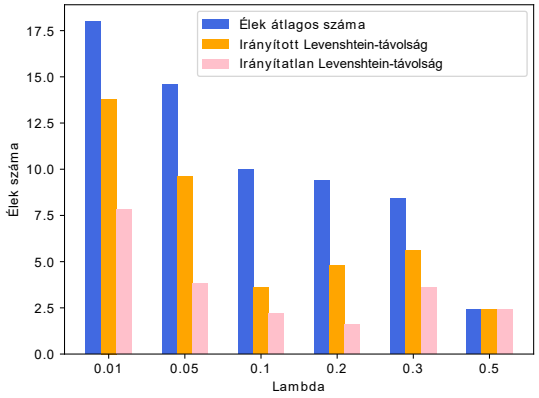
Fontos kiemelnem még, hogy a NOTEARS algoritmus nem garantálja az élek megfelelően irányított behúzását: csupán az ekvivalencia osztály egy tagját találja meg az algoritmus. A dolgozatomban ugyan az irányított élek távolságát is vizsgltam, azonban nem érdemes túl nagy jelentőséget tulajdonítani ezeknek az értékeknek, elsősorban az irányítatlan gráfok közötti távolságot érdemes figyelembe venni

Az 3.6 ábrákon két különböző nagy mintaszámú partner adataiból épített gráfok különbségei láthatók: n=1000 mintaszámú bootstrap adathalmazokon futtattam az algoritmust különböző λ paraméterekkel mindkét partner esetén, majd 5 ilyen futás által generált gráf távolságát hasonlítottam össze. Az ábrákról sok minden leolvasható: egyrészt, hogy az algoritmus általánosan jobban teljesített az I kódú partner, mint az E kódú parnter esetében. Ez valószínűleg azt jelenti, hogy az E partner adathalmaza kevésbé homogén (ez például lehet azért, mert sok komplikált eset kerül oda).

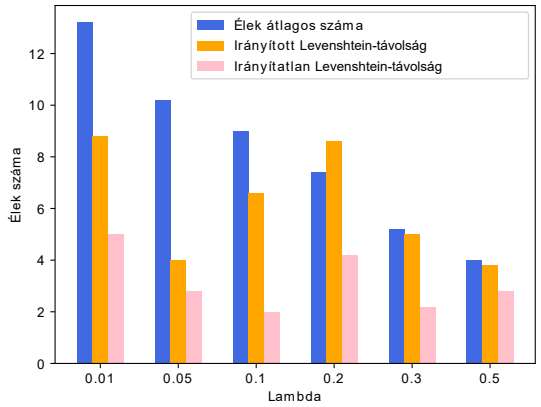
Látszik az is, hogy $\lambda = 0.5$ esetben már nagyon kevés élet rajzol be az algoritmus, és nagyon rosszul is teljesít. $\lambda = 0.01$ esetben nagyon sok élet behúz, ugyanakkor ezeket



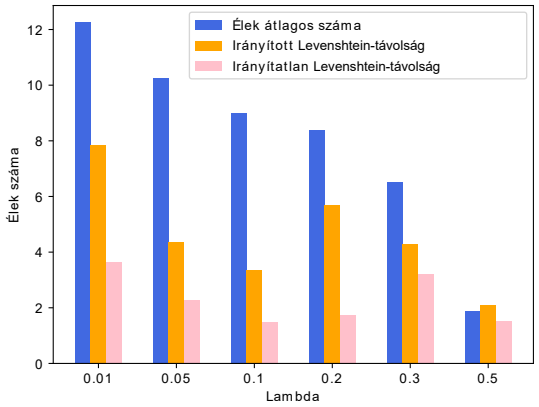
(a) E partner adataiból, $\omega = 0.1$ paraméterrel



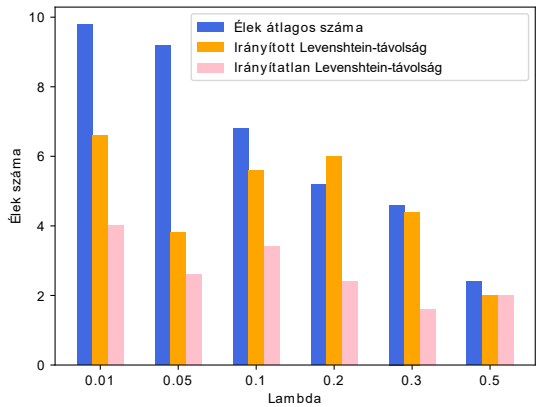
(b) I partner adataiból, $\omega = 0.1$ paraméterrel



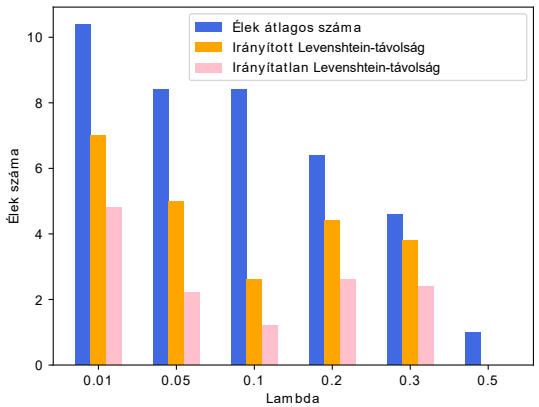
(c) E partner adataiból, $\omega = 0.2$ paraméterrel



(d) I partner adataiból, $\omega = 0.2$ paraméterrel



(e) E partner adataiból, $\omega = 0.3$ paraméterrel



(f) I partner adataiból, $\omega = 0.3$ paraméterrel

3.6. ábra. Két nagy mintaszámú partner (E és I kódú partnerek) adataiból különböző ω értékekkel a NOTEARS algoritmussal épített gráfok átlagos élszáma és Levensthein távolsága.

meglehetősen nagy arányban nem is találja el. Ez még nagy ω esetén is igaz, tehát érdekes módon a vágási küszöb sem javít a találatok arányán.

A $\frac{LT}{e}$ hányadosokat kiszámítva (ahol e az élek számát jelöli) ugyanaz jön ki, mint ami a grafikonokról is leolvasható: a $\lambda = 0.05$ és $\lambda = 0.1$ paraméterekkel generált gráfok hasonlítanak legjobban egymásra. (Az, hogy pontosan melyik a jobb, függ attól, hogy az irányítatlan vagy az irányított gráfok közti Levenshtein-távoságot tekintjük mérvadónak.) Ez is leginkább $\omega = 0.2$ esetén igaz.

Így a $\lambda = 0.1$ és $\omega = 0.2$ körüli paraméterválasztások bizonyulnak a legmegfelelőbbnek.

Utolsó lépésként vizsgáltam meg az algoritmus c lépésköz paraméterének és α_0 Lagrange-multiplikátor kezdőértékének beállítását. c paraméter ott játszik szerepet, amikor egy iteráció során W_t értékének frissítésekor csak olyan W_{t+1} kerül elfogadásra, amely teljesíti, hogy $h(W_{t+1}) < ch(W_t)$. Vagyis nem elegendő egy olyan W_{t+1}-t találni, amelyre $h(W_{t+1})$ éppen kisebb, mint $h(W_t)$, megkövetelünk bizonyos mértékű javulást. Ha $h(W_{t+1})$ értéke nem csökkent kellően $h(W_t)$-hez képest, ρ értékének növelésével folytatódik az algoritmus új minimumkereséssel.

Érdekes módon a publikált cikkben [16] nem említik külön a c paraméter beállítását, az algoritmus implementációjában a kódba beleírva található a $c = 0.25$ kezdeti érték. Nyilván c értéke 0 és 1 között mozoghat. Én az alábbi értékeket próbáltam ki: $c \in \{0.01, 0.05, 0.1, 0.2, 0.25, 0.3, 0.5\}$ A már ismert keretek között teszteltem, nagyobb mintaszámú partnerek adathalmazán futtattam az algoritmust $\lambda = 0.1$, $\omega = 0.2$ és c paraméter különböző értékei mellett. Nagy különbségek nem voltak megfigyelhetők a gráfok között, a 7 futási eredmény általában $c = 0.01, c = 0.05$ esetleg $c = 0.1$ esetén különbözött a többi gráftól, melyek izomorfak voltak. Ez talán nem is meglepő, c paraméter ilyen alacsony értéke mellett ugyanis a kiterjesztett Lagrange-típusú módszer előnyös tulajdonságai kevésbé kihasználhatóak. Illetve egy centrum esetében megfigyelhető volt $c = 0.3$ és $c = 0.5$ esetén még egy él behúzása a kisebb c paraméterekkel generált gráfokhoz képest. Ezek alapján $c = 0.25$ paraméter érték egy jó választásnak tűnik.

α_0 a kiterjesztett Lagrange-típusú módszer Lagrange-multiplikátorának kezdeti értéke. Az algoritmusban eredetileg az $\alpha_0 = 0$ inicializálást alkalmazzák. Biztosan nem érdemes α_0-nak nagy kezdőértéket beállítani, hiszen éppen az a paraméter lényege, hogy iterációnként növekedjen az értéke. Nagyobb partnerek teljes adathalmazán c paraméterhez hasonlóan α_0-t tesztelve $\lambda = 0.1$ és $\omega = 0.2$ paraméterek mellett $\alpha \in \{0.01, 0.1, 0.2, 0.3, 0.5, 0.7, 0.9\}$ értékekre az volt látható, hogy α_0 kezdeti értéke nem igazán befolyásolja a generált struktúrákat. Volt olyan partner, akinek a struktúrája teljesen fix maradt α_0 minden értéke mellett, volt, ahol nagyobb α_0 értékeknél 1-2 élkülönbség megjelent. Összességében azonban α_0 kezdeti értékéről is azt lehet elmondani, hogy nagyban nem befolyásolja az algoritmus futását, ezért megfelelő az $\alpha_0 = 0$ érték választása.

A 3.2 számozású táblázat tartalmazza összefoglalva a NOTEARS algoritmust illetően a paramétervizsgálataim eredményét.

Paraméter	λ	ω	max_iter	loss_type	ρ_{max}	c	α_0
Érték	0.1	0.2	100	l2-loss	10^{16}	0.25	0

3.2. táblázat. A NOTEARS algoritmus legjobbnak talált paraméterbeállításai összefoglalva

3.3.2. NOTEARS-ADMM algoritmus

A NOTEARS-ADMM algoritmus működése a nevével ellentétben nem áll annyira közel a NOTEARS algoritmuséhoz. A paraméterei ugyan hasonlóak, de fontos hangsúlyozni, egy

federált DAG-tanulási algoritmusról van szó, ahol az egyes partnereknél, majd a központi egységen is végrehajtásra kerülnek lépések.

Mivel a két algoritmus paraméterezése nagyon hasonló, könnyen azt hihetnénk, hasonló paraméterbeállítások mellett a két algoritmus kimenete is hasonló lesz. Ez azonban - mint látni fogjuk - közel sincs így minden esetben: a fent említett oknál fogva, a NOTEARS-ADMM egészen más hiperparaméter beállítást, így önálló vizsgálatot igényelt.

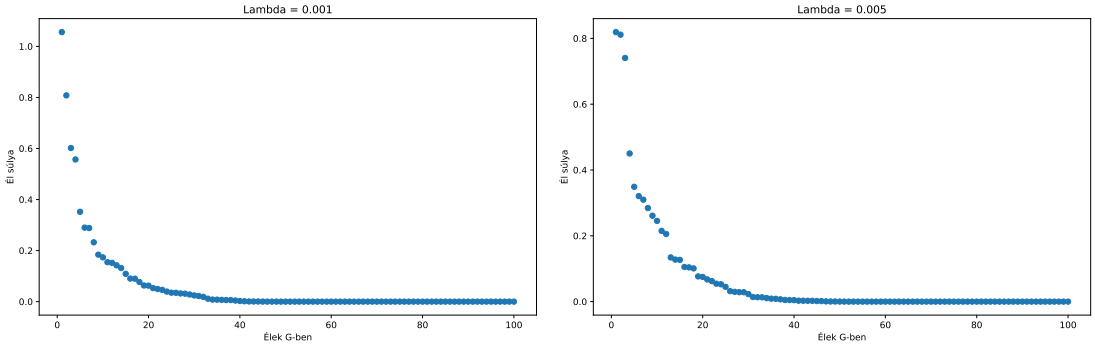
A NOTEARS-ADMM az alábbi állítható hiperparaméterekkel rendelkezik:

- λ: ℓ_1 reguarizáció paramétere, ugyanaz a szerepe, mint a NOTEARS algorimus esetén
- ω: a vágási paraméter, jelentése ugyanaz, mint a NOTEARS algoritmus esetén
- ρ_1 és ρ_2: büntető paraméterek kezdeti értékei az algoritmusban
- ρ_{max}: ha ezt az értéket eléri ρ_1 és ρ_2 is, az algoritmus leáll
- max_iter: az a maximális iterálási szám, amit ha túllép az algoritmus, leáll. Figyeltem a paraméter értékét a futtatások során, de nem adódott gond abból, hogy az iterációk száma elérte volna ezt az értéket, így nem volt szükséges változtatnom, az eredeti max_iter=200 értéken hagytam.
- γ_1: azt kontrollálja, ρ_1 milyen ütemben növekedjen
- γ_2: azt kontrollálja, ρ_2 milyen ütemben növekedjen
- $\beta_1^1, \beta_2^1, ...\beta_k^1, \alpha$: ezek a Lagrange-multiplikátorok becslései. Kezdeti értéküknek kevésbé van jelentősége, mivel az algoritmus minden iteráció során frissíti őket. Így én is a tanulmány által javasolt beállítást használtam, mindegyiket 0-ra, illetve a β-k esetében csupa 0 mátrixra inicializáltam.

Valamivel több paraméter állítása lehetséges, mint a NOTEARS algoritmus esetén. Itt még inkább igaz volt, hogy nem lehetett szigorúan, egymás utáni sorrendben megtalálni a paraméterek megfelelő értékeit, hanem iteratív módon többször vissza kellett térnem, és felülvizsgálnom egy korábbi paraméter beállítását. Kezdeti próbafuttatások során megfigyelhető volt, hogy leginkább a λ, az ω, a γ_1 és γ_2 paraméterek beállítása befolyásolja az algoritmus kimenetét (a NOTEARS esetéhez hasonlóan). Így az alapvető stratégiám az volt, hogy először a λ és az ω értékét korlátozom egy megfelelőnek tűnő tartományra, majd ezek mellett vizsgálom az algoritmus többi paraméterének értékét. Szükség esetén ezek után újra állítok λ és ω értékein.

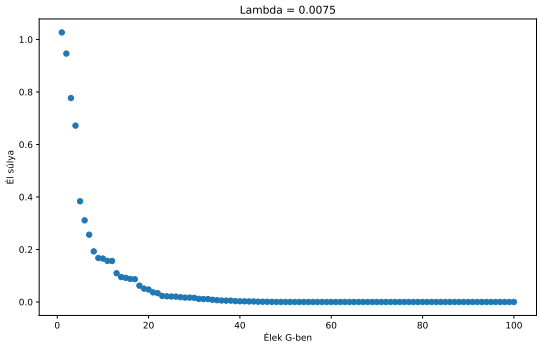
A NOTEARS algoritmus paraméterezésénél leírtakhoz hasonlóan a NOTEARS-ADMM esetén is azzal kezdtem, hogy $\omega = 0$-s próbafuttatások során megnéztem, milyen tartományban mozognak a behúzott élek súlyai. A 3.7-es ábrán a vizsgált λ értékekkel való futtatások eredményei láthatók (azt, hogy miért ezeket a λ értékeket használtam, alább részletezem).

Alapvető különbésg a NOTEARS-ADMM és a NOTEARS algoritmus kimenete között, hogy míg a NOTEARS algoritmus egy jó paraméterbeállítás mellett kevés véletlen zajt tartalmaz, addig a NOTEARS-ADMM kimeneti gráfjában sok nagyon kicsi (10^{-3} és annál kisebb nagyságrendű) élsúlyok szerepelnek. Ez megfigyelhető a 3.7 és a 3.4, illetve a 3.5 grafikonok összehasonlításával: a 3.4-es és a 3.5-ös ábrákon a NOTEARS esetében egy meredekebb emelkedés tapasztalható a súlyok tekintetében. Ez azért van, mert nagyon sok nulla súlyú él szerepel az eredménygráfban, ezeken kívül pedig aránylag nagy abszolútértékű a többi megjelenő él. A NOTEARS-ADMM esetében pedig, ahogy említettem, nem nulla, hanem nagyon kis abszolútértékű élek jelennek meg, emiatt fokozatosabb az emelkedés az élek súlyát tekintve a 3.7-es ábrán.

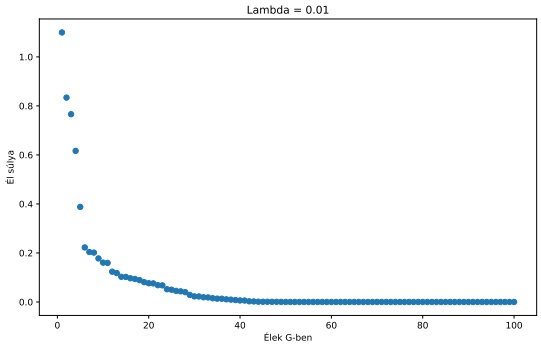


(a) $\lambda = 0.001$

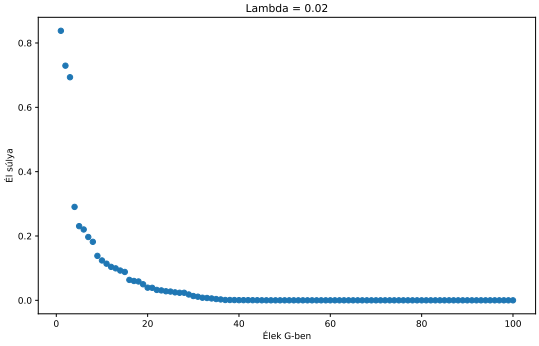
(b) $\lambda = 0.005$



(c) $\lambda = 0.0075$



(d) $\lambda = 0.01$



(e) $\lambda = 0.02$

3.7. ábra. A vágatlan élek súlyainak alakulása a globális adatból n=5000 mintaszámú bootstrap halmazon futtatva az algoritmust az adott λ paraméterekkel.

Megfigyelhető még, hogy a λ paramétertől függően néhány esetben van egy nagyobb törés az élek súlyának emelkedésében 0.2 körül. Azonban ez megint nem általánosan igaz minden futattásra. Amit mégis a 3.7 ábra alapján el lehet mondani, nagyon hasonló a NOTEARS algoritmusnál megállapítottakhoz: ω paramétert 0.1 és 0.3 közé érdemes beállítani. 0.1 alá nem érdemes menni, mert abban a tartományban kevésbé meredeken emelkedik a görbe, könnyen belekerülhetnek a zajból származó összefüggések a modellbe. 0.3 súly fölött pedig nagyon kevés él található.

Megfigyeltem különböző λ paraméterek használata esetén az algoritmus kimenetét: nagyon érdekes eredmény volt, hogy a $\lambda \geq 0.05$ esetén az algoritmus már abszolút nem működött jól: már egyetlen 10^{-1} nagyságrendű élet sem húzott be, azonban az akár 10^{-14} nagyságrendű élek száma jelentősen megnövekedett. Ha $\lambda = 0.02$, ez a hatás még nem jelent meg, így ez lett a legnagyobb λ paraméterérték, amit vizsgáltam. Ha a kicsi λ értékekekt vizsgáljuk, ott az algoritmus nem mutat ilyen anomáliákat. A $\lambda = 0.001$ egy észszerű alsó korlátja a vizsgált λ paramétereknek.

A ρ_{max} paraméter értékét vizsgálva azt lehetett észrevenni, hogy nagyon tág tartományon belül nem befolyásolja az algoritmus kimenetét: a tanulmányban a $\rho_{max} = 10^{16}$ értéket javasolják, én azt figyeltem meg, hogy a generált gráf akkor egészen addig nem változott, amíg ρ_{max} értékét 10^6-ra nem csökkentettem. A felső határt illetően még 10^{40} esetén is ugyanazt a kimenetet adta az algoritmus, így ρ_{max} paraméter értékét nem változtattam meg, meghagytam $\rho_{max} = 10^{16}$ értéken.

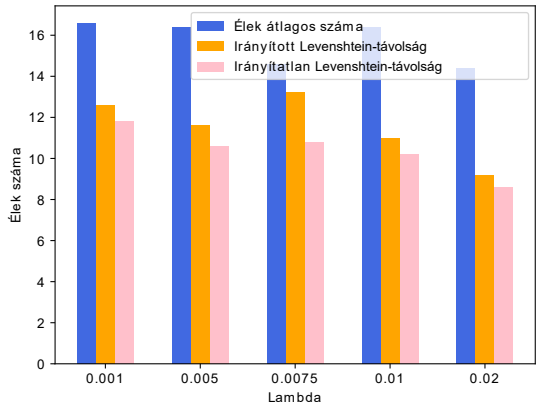
A ρ_1 és ρ_2 paraméterek állítása is csak kis mértékben befolyásolta az algoritmust, hiszen ezek kezdeti értékek, valójában ρ_1 és ρ_2 értéke minden iteráció során frissül. Ez pedig már γ_1 és γ_2 által szabályozható.

Ezután a γ_1 és γ_2 paraméterek beállításával folytattam. Ez után a lépés után többször állítanom kellett még a λ és ω paramétereken, mert a γ_1 és γ_2 paraméterek értéke az egész algoritmus működésére kihatott. Ezek a paraméterek felelősek ρ_1 és ρ_2 frissítéséért. A tanulmányban [12] a szerzők $\gamma_1 = 1.75$ és $\gamma_2 = 1.25$ értékeket használtak, azonban arról nem közöltek statisztikát, hogy mi alapján választották ezeket az értékeket. Ezen kezdőértékek mentén indultam el: a legbiztosabb vizsgálati pontnak most is az tűnt, ha megfigyelem, rögzített λ és ω, illetve különböző γ_1 és γ_2 értékek mentén hogyan alakul az algoritmus kimenete. A korábban részletezett futásokhoz hasonló módon, n=500 mintaszámú, visszatevéses mintavétellel választott bootstrap adathalmazokon a már bemutatott metrikák szerint vizsgáltam az algoritmus kimenetét, a grafikonokon az élek száma mellett az irányított és irányítatlan esetbeli Levenshtein-távolságok olvashatók adott γ_1, γ_2 paraméterválasztással.

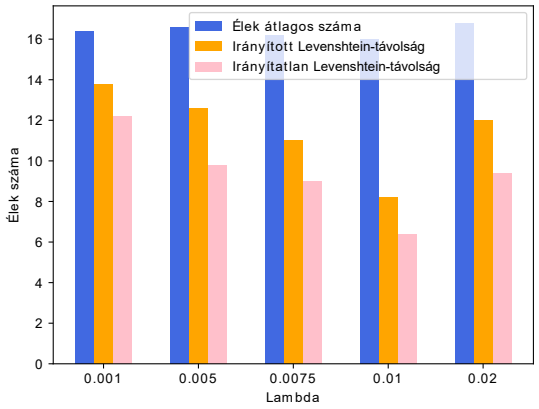
A tanulmányban rögzített $\gamma_1 = 1.75$ és $\gamma_2 = 1.25$ értékekből kiindulva a futtatások során az alábbi értékeket párosítottam γ_1 és γ_2 paraméterekhez:

$$\gamma_1, \gamma_2 \in \{1.1; 1.25; 1.5; 1.75; 2; 5\}$$

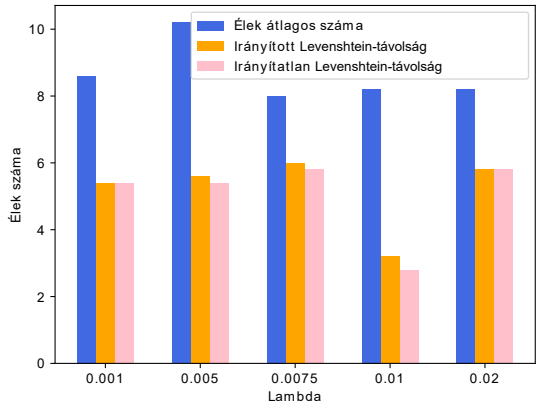
Mivel ρ_1 és ρ_2 értéke a 2.21 és 2.22 alapján kerül frissítésre minden iterációban, ezért nyilván nincs értelme az 1-et vagy egy annál kisebb számot paraméterként használni. Valószínűleg az 5 érték is túl nagy lesz már, azonban mint szélsőséges értéket, szándékosan bent hagytam a tesztben. Ezekeből az értékekből képzhető összes lehetséges γ_1-γ_2 értékpárra vizsgáltam az algoritmus kimenetét különböző λ értékek mellett. (Ezeknek a futásoknak az eredményeit a terjedelemre való tekintettel nem szemléltetem külön ábrával.) Az esetek nagy részében az algoritmus nem megfelelő eredményt generált, általában azért, mert csak nagyon lassan konvergált az optimum felé. Végül a 36 féle paraméterpár-kombinációk közül kettő teljesített a legjobban, ezek: $\{\gamma_1 = 1.75; \gamma_2 = 1.25\}$ és $\{\gamma_1 = 2; \gamma_2 = 1.25\}$ értékek. Ennek a két paraméterbeállításnak láthatók a futási eredményei különböző λ és ω paraméterek mellett a 3.8-as árbán.



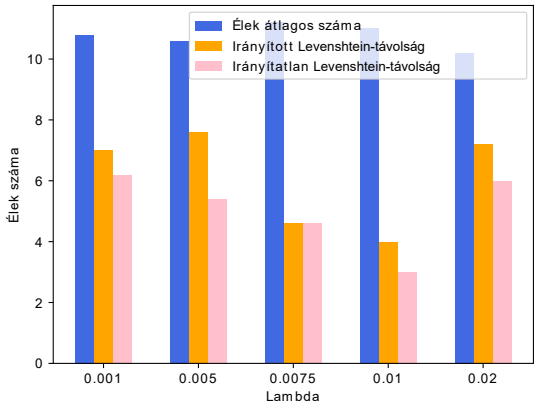
(a) $\gamma_1 = 1.75, \gamma_2 = 1.25, \omega = 0.1$



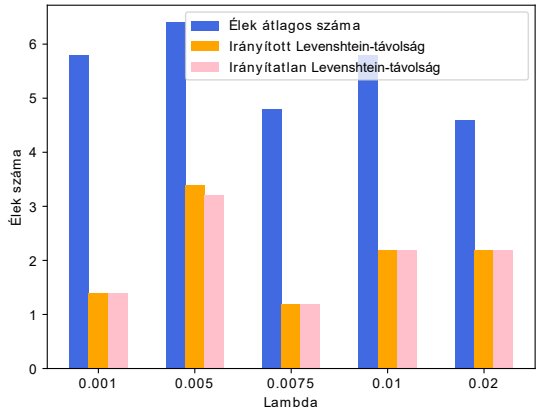
(b) $\gamma_1 = 2, \gamma_2 = 1.25, \omega = 0.1$



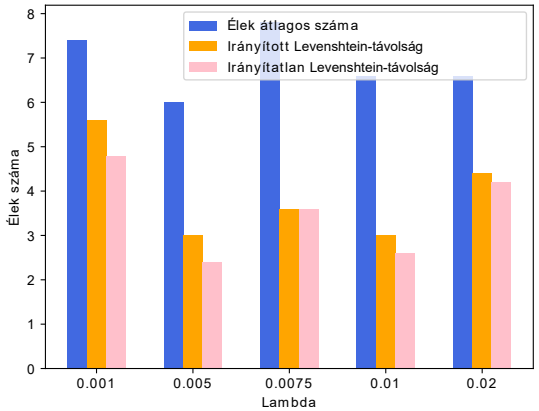
(c) $\gamma_1 = 1.75, \gamma_2 = 1.25, \omega = 0.2$



(d) $\gamma_1 = 2, \gamma_2 = 1.25, \omega = 0.2$



(e) $\gamma_1 = 1.75, \gamma_2 = 1.25, \omega = 0.3$



(f) $\gamma_1 = 2, \gamma_2 = 1.25, \omega = 0.3$

3.8. ábra. A $\{\gamma_1 = 1.75; \gamma_2 = 1.25\}$ és a $\{\gamma_1 = 2, \gamma_2 = 1.25\}$ paraméterpárok esetén a futtatások eredménye különböző λ és ω paraméterek mellett.

A 3.8-as grafikonokról leolvasható, hogy ha kevés behúzott élet szeretnénk kapni az eredménygráfban, $\gamma_1 = 1.75, \gamma_2 = 1.25$, $\omega = 0.3$, $\lambda = 0.001$ körüli paraméterértékeket érdemes választani. Ha szeretnénk azonban kevésbé nyilvánvaló összefüggéseket is megkapni, de azok közül is csak a biztosabbakat, $\gamma_1 = 2, \gamma_2 = 1.25$, $\omega = 0.2$, $\lambda = 0.01$ körüli paraméterértékek a legmegfelelőbbek. Én az utóbbit használtam a dolgozatom következő fejezetében, egyrészt, mert szerettem volna néhány kevésbé biztos kapcsolatot is megkapni, másrészt, mert a NOTEARS algritmus paraméterezése ilyen módon nagyon hasonló.

Összességében tehát a 3.3 táblázatban rögzített paraméterbeállításokat fogom használni a NOTEARS-ADMM algoritmus futtatása során a dolgozatomban.

Paraméter	λ	ω	max_iter	γ_1	γ_2	ρ_{max}	ρ_1	ρ_2	β_k	α
Érték	0.01	0.2	200	2	1.25	10^{16}	0.001	0.001	0	0

3.3. táblázat. A NOTEARS-ADMM-hez használt paraméterek rögzítve

4. fejezet

Eredmények

Ebben a fejezetben a már helyesen felparaméterezett algoritmusook futási eredményeit mutatom be és hasonlítom össze.

4.1. Globális és lokális gráfok építése

4.1.1. Globális gráf

A megfelelően felparaméterezett algoritmust a globális adaton futtatva próbáltam ki először. A NOTEARS esetén a két fő hiperparaméter, a λ és az ω néhány jónak ítélt paraméterbeállításával generált gráfok láthatók a 4.1-4.3 ábrákon. A futtatások ugyanazon a kipótolt adaton lettek generálva, ezekben az esetekben a kipótlás nem okozhatja a gráfok közötti különbségeket. A λ paraméter 0.05 és 0.1 közül veszi fel az értékét, ω paraméter pedig 0.1 és 0.2 közül a 3.3.1 alfejezetben tett megállapításoknak megfelelően. Látható, hogy a $\lambda = 0.05, \omega = 0.2$ és $\lambda = 0.1, \omega = 0.2$ beállításokkal ugyanaz a gráf generálható. A másik két esetben az algoritmus jóval több élet behúz. Ez is a 3.3.1 alfejezetben tett megállapítás erősíti, hogy a $\lambda = 0.1$ és $\omega = 0.2$ körüli paraméterezésekre viselkedik legrobusztusabb módon az algoritmus.

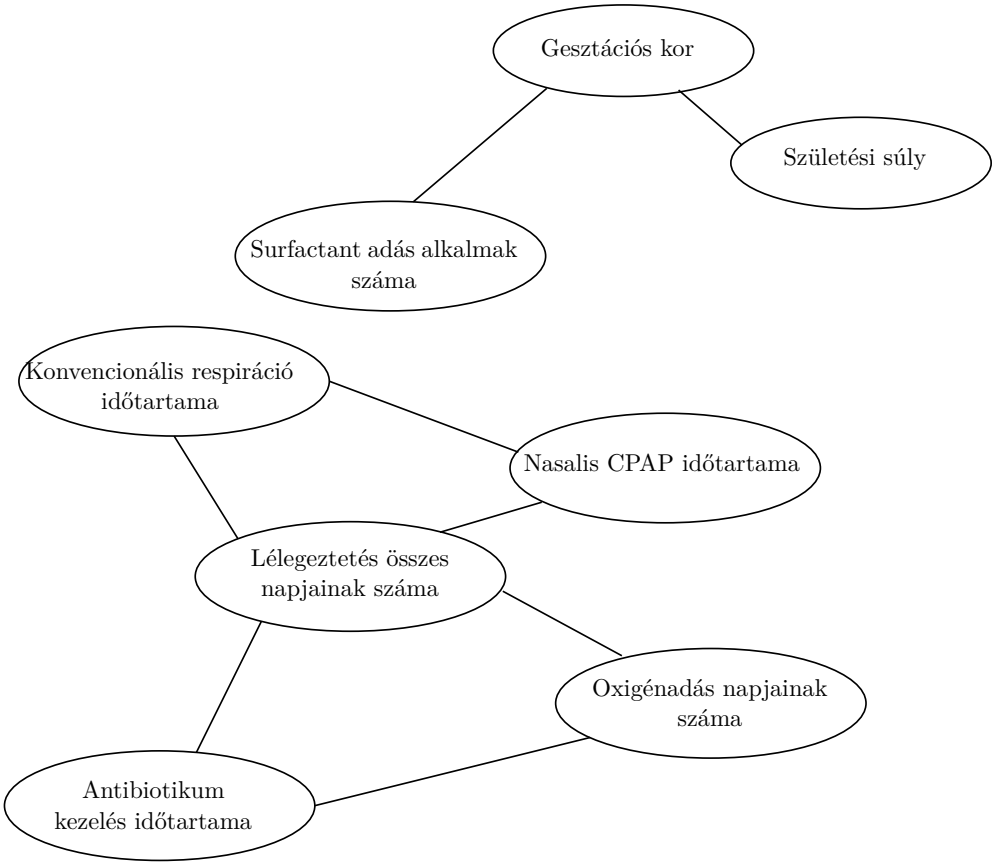
A behúzott éleket tekintve a triviális összefüggéseket mindegyik esetben behúzta az algoritmus. Ezek:

- A *Lélegeztetés összes napjainak a száma* egy képzett adat: a *Nasalis CPAP időtartama*, a *Konvencionális respiráció időtartama* és még egy harmadik, a gráfban nem szerpelő változó összegeként számítódik ki.
- *A Születési súly* és a *Gesztációs kor* között is elvárt, hogy kapcsolat legyen.

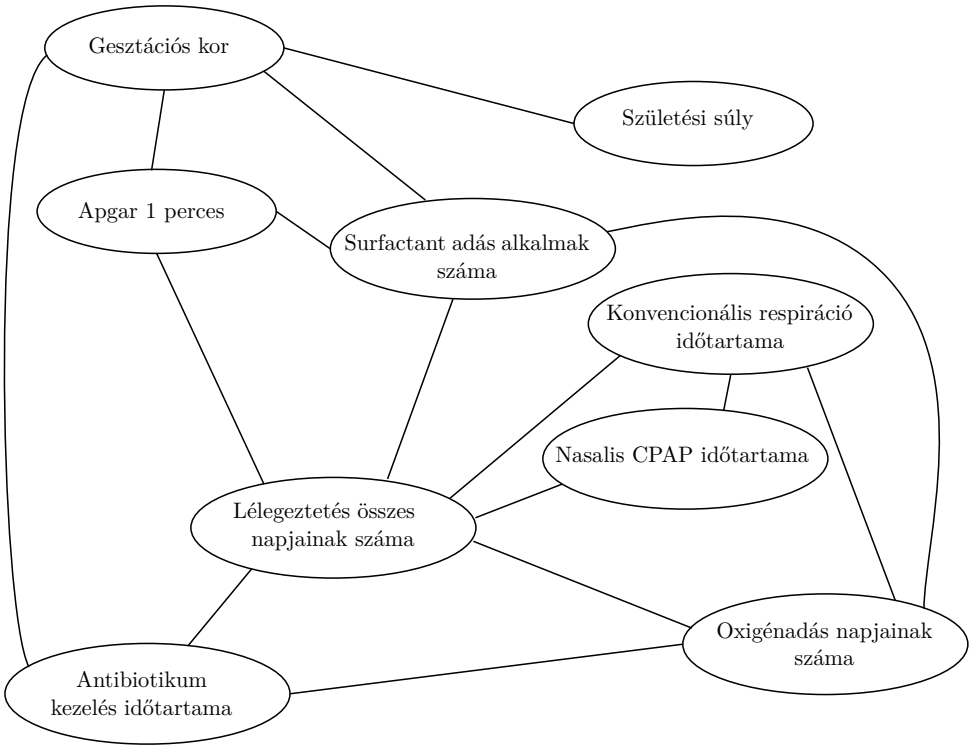
4.1.2. Lokális gráfok építése a partnerek adataiból

Az egyes partnerek adataiból épített gráfoknak is nagy jelentősége van: ezek vizsgálatával deríthető ki, ha egy partner valamiben eltér a többiektől, lehetőséget nyújtva ezzel a visszajezésre és a lokális beavatkozásra, legyen szó akár az adatrögzítés módjáról, akár a döntési protokollokról.

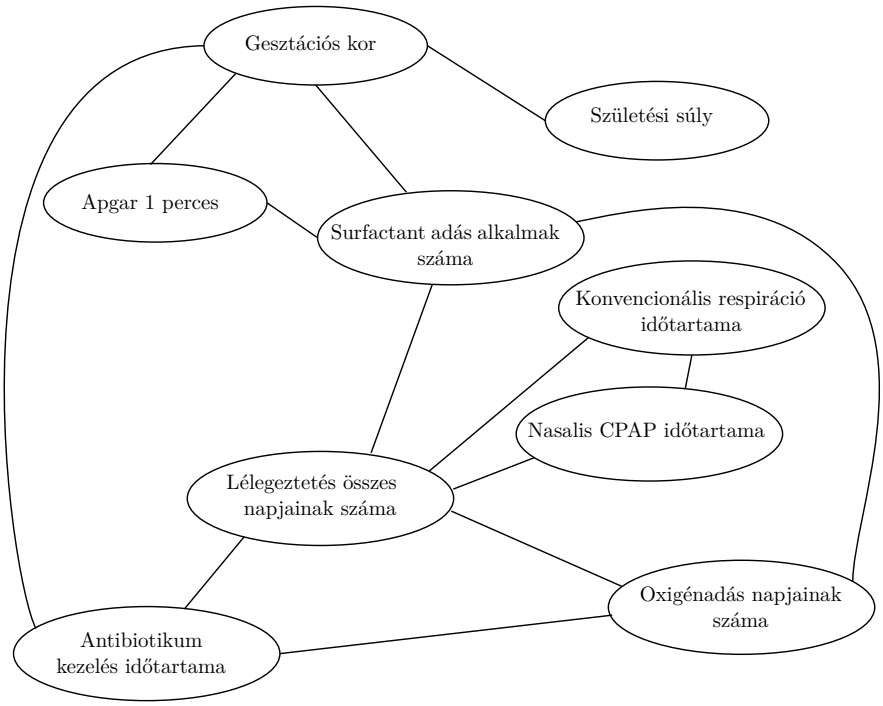
Nem triviális azonban, hogy az egyes centrumok strukúráinak különbségeit milyen metrika szerint vizsáljuk. Egy lehetőség azt vizsgálni, hogy a globális struktúrától mennyi az egyes centrumok Levenshtein-távolsága (ezt a megközelítést vizsgálom is a 4.1.3 alfejezetben). Ekkor azonban a nagyobb centrumok mintaszámuk miatt sokkal nagyobb súllyal számítódhatnak a legkisebb centrumokhoz képest. Ezért máshogyan kell azt az információt kezelni, ha egy nagy mintaszámú, illetve egy kis mintaszámú partner globális gráftól való eltérését vizsgáljuk.



4.1. ábra. A teljes adathalmazon épített gráf $\lambda = 0.05$ és $\omega = 0.2$ paraméterezéssel, illetve gráf $\lambda = 0.1$ és $\omega = 0.2$ paraméterezéssel



4.2. ábra. A teljes adathalmazon épített gráf $\lambda = 0.05$ és $\omega = 0.1$ paraméterezéssel



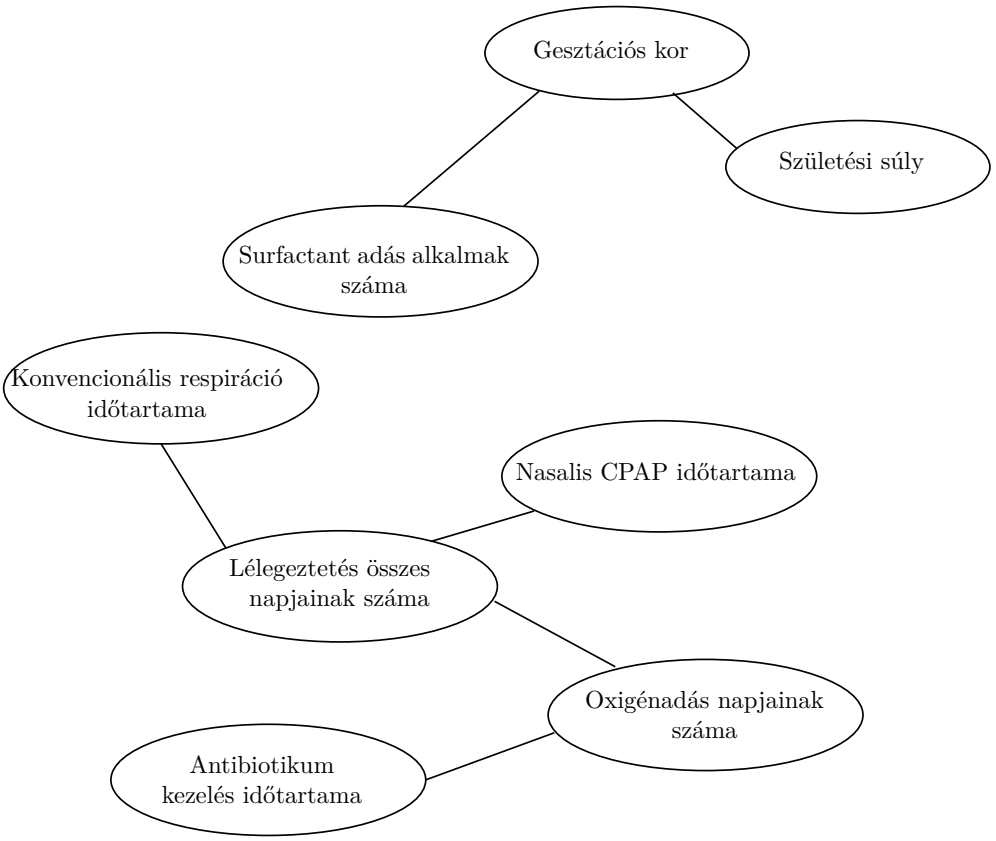
4.3. ábra. A teljes adathalmazon épített gráf $\lambda = 0.1$ és $\omega = 0.1$ paraméterezéssel

Ha azonban a partnerek adataiból egyenként, lokálisan épített gráfoknak a konszenzus-gráfját képeznénk és az ettől való eltérést számítanánk ki, szintén egy vizsgálható képet kapnánk arról, az egyes partnerek struktúrái mennyire hasonlóak, van-e olyan esetleg, amelyik nagyon eltér a többitől.

A 7. definíció alapján képzett konszenzus-gráfnak az egyes partnerek adataiból épített gráfoktól való eltérését már tudjuk számszerűsíteni, a már bevezetett Levenshtein-távolság formájában. Érezhető, hogy ez sok esetben nem fog tökéletes eredményt hozni: a vágás miatt például egy 0.4 és egy 0.6 valószínűséggel előforduló él közül a 0.6-os bekerül, a 0.4-es nem a konszenzus gráfba. Ez azt eredményezheti, hogy előfordulhat egy olyan G_i gráf, amiben sok olyan él szerepel, ami a $G_1, G_2, ...G_N$ gráfokban csak 0.6 valószínűséggel fordul elő, tehát éppen több, mint az esetek felében van behúzva. Ha előfordul egy olyan G_j gráf is, amiben sok olyan él szerepel, ami éppen kevesebb, mint a felében fordul elő a $G_1, G_2, ...G_N$ gráfoknak, G_j gráf a konszenzus gráftól vett távolsága lényegesen nagyobb lehet, mint G_i távolsága, miközben valójában nem olyan nagy a különbség közöttük.

Sajnos, amennyiben az éleket súlyok nélkül kezeljük, nem adható olyan megoldás, amivel ezek a különbségek feloldhatóak lennének: bárhol vágunk, lesznek élek éppen a vágási feltétel alatt és felett. Az optimális vágási küszöböt a különböző típusú hibák, mint hibás felfedezés és elmulasztott felfedezés költségei határozzák meg, amelyek egyenlősége esetén a 0.5 az optimális. (Természetesen, ha vágatlan formában, súlyozott élekkel kezelnénk a gráfokat, ott másfajta nemkívánt viselkedés lépne fel: ott az torzítaná az eredményt, hogy például ha egy gráf 0.95 súllyal tartalmaz egy élt, ami az átlagban csak 0.8 súllyal fordul elő, a gráf távolságába ez a 0.15 is beleszámítana. Közben pedig józan ésszel belegondolva, a konszenzus és az egyedi gráfban is nagyon erős formájában jelen van ez az él, nem szeretnénk, hogy sok ilyen kicsi eltérés miatt legyen egy gráf konszenzus-gráftól mért távolsága nagy.)

Rátérve az eredményekre: a 4.4 ábrán látható a partnerek lokális struktúráiból képzett irányítatlan éleket tartalmazó konszenzus-gráf. Konszenzus-gráf az irányított élek mentén is számolható, azonban mivel a NOTEARS algoritmus csak az ekvivalencia osztályokat



4.4. ábra. Irányítatlan konszenzus-gráf

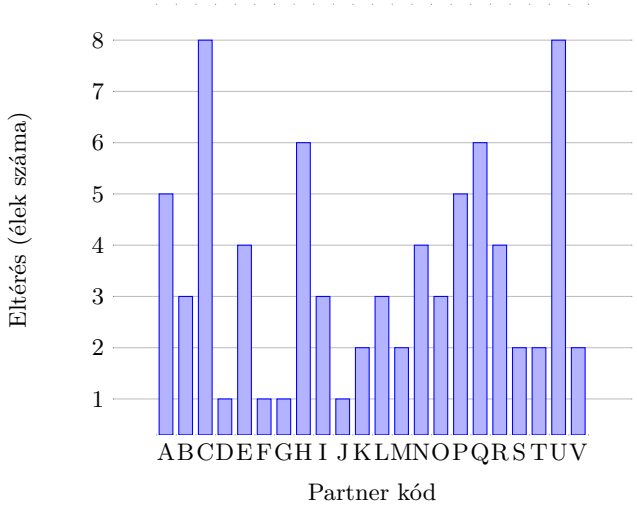
találja meg az adatban, sokat hibázik az irányított élek mentén, így azt a gráfot nem érdemes vizsgálni.

Az irányítatlan gráfban az alábbi összefüggések jelentek meg:

- A *Születési súly* nyilván nagyon erősen függ a *Gesztációs kortól*
- A *Lélegeztetés napjainak számáról* tudjuk, hogy a *Nasalis CPAP időtartamából*, a *Konvencionális respiracio időtartamából* (és egy harmadik, nem jelenlévő változóból) képzett adat
- Az *Oxigénadás időtartama* is nyilvánvalóan összefügg a *Lélegeztetés napjainak számával*
- A *Gesztációs kortól* függhet a *Surfactant adás alkalmak száma*
- Az *Antibiotikum adások száma* és az *Oxigénadás időtartama* összefüggés már érdekesebb kapcsolat, már szakmai ismeretek szükségesek az él helyességének igazolásához

Összességében viszonylag kevés, csak a legbiztosabb kapcsolatokat rajzolta be az algoritmus. Ha megnézzük az egyes partnerek struktúráit, azok átlagosan 8.1 élet tartalmaznak, míg az irányítatlan konszenzus gráf 6-ot, (az irányított konszenzus gráf egyébként csak 4-et, de ezt nem vizsgáltam). Ebből azt a következtetést lehet levonni, hogy az átlagosan 8 élnek a negyede kevesebb, mint a gráfok 50%-ában fordul csak elő. Tehát sok olyan különböző él van, amelyiket kevesebb, mint a partnerek felénél talál meg az algoritmus.

Ha már megvan az irányítatlan konszenzus gráf, ki lehet számolni az egyes partnerek egyéni eltéréseit ettől. A 4.5-ös ábrán minden centrumra az összesített eredmény látható: az azonosító kódjuk és az irányítatlan konszenzus-gráftól vett Levenshtein-távolságuk.



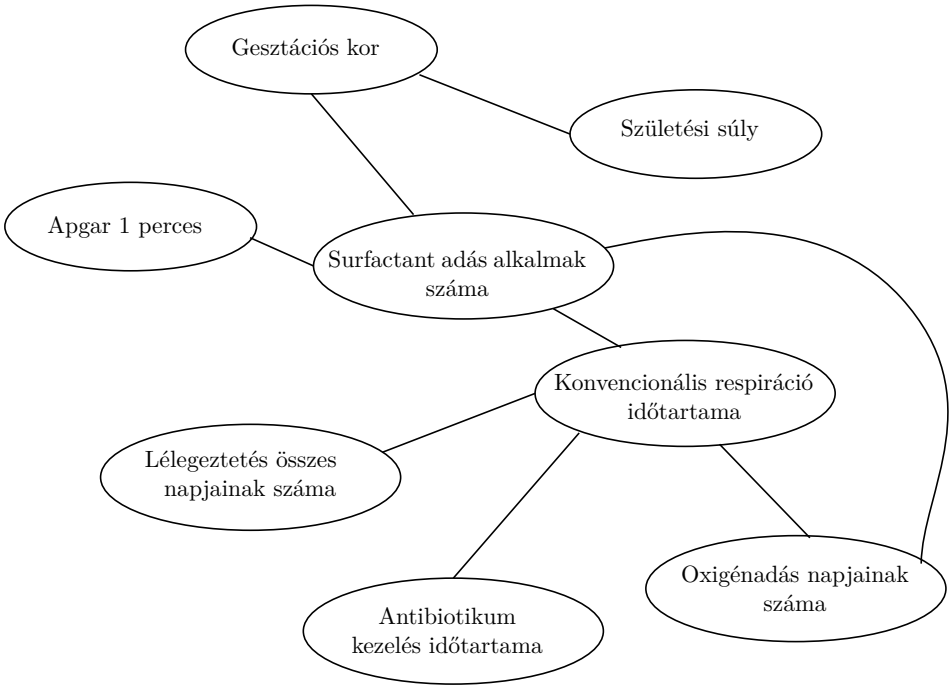
4.5. ábra. A partnerek lokális struktúrának különbözősége a konszenzus gráfoktól irányítatlan esetben

A 4.5-ös ábráról az alábbiak olvashatók le:

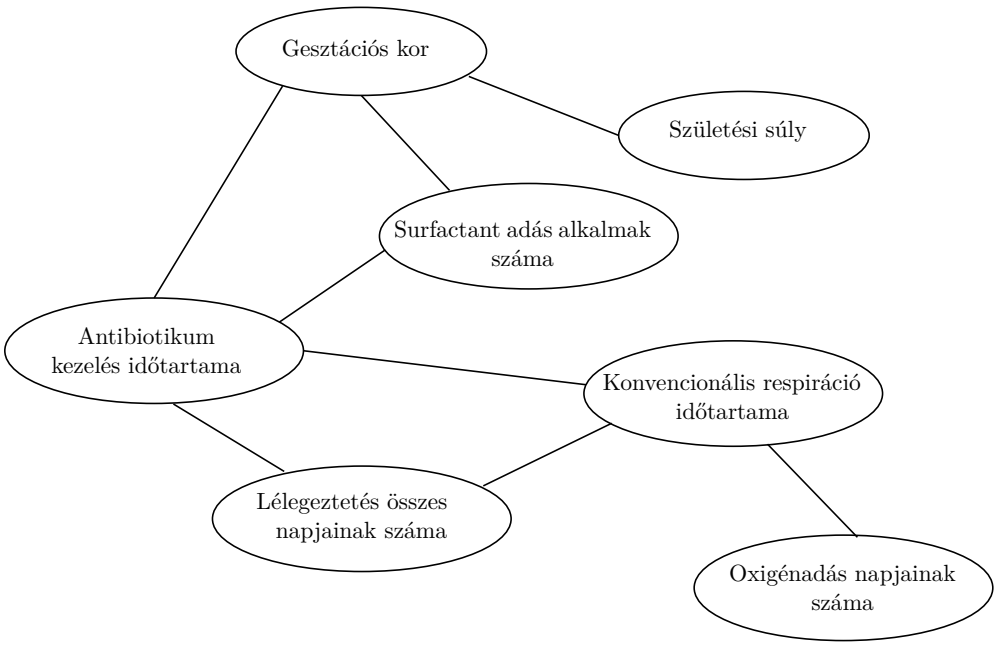
1. Az átlagos Levenshtein-távolság irányítatlan esetben 3.5 él.
2. A legkisebb távolság értéke négy centrum kódja mellett is szerepel, ezek a D, F, G, és a J kódú centrumok. A struktúráikat megvizsgálva megállapítható, hogy az irányítatlan gráfhoz képest mind 1-1 plusz élet tartalmaznak.
3. Az U és a C kódú centrum mellett szerepel a legnagyobb különbség, mely 8 élbeli különbésget jelent. A lokális struktúrák a 4.6-os és 4.7-es ábrákon láthatók. Egyik sem találta meg az alábbi, az konszenzus gráfban szereplő összefüggéseket:
   (a) A *Lélegeztetés napjainak száma* és a *Konvencionális respiráció időtartama* közötti élet
   (b) A *Lélegeztetés napjainak a száma* és az *Oxigénadás alkalmak száma* közötti élet
   (c) Az *Antibiotikum kezelés ideje* és az *Oxigénadás alkalmak száma* közötti élet

Ezek közül az (a) pont, ami mindenképp problémás, hiszen itt származtatott adatról van szó. De ezeken az éleken túl is meglehetősen különbözik a struktúra az átlagosaktól. Az U kódú centrum esetén külön érdekes, hogy a *Surfactant adás alkalmak számát* nagyon sok másik változóval összekötötte az algoritmus. Szakmai szempontból érdekes lehet ennek a vizsgálata.

Összességében elmondható, hogy a lokális struktúrák meglehetősen különbözőek. Kevés élet vizsgálva is a centrumok körülbelül fele 2-nél több élben eltér a konszenzus-gráftól irányítatlan esetben. Az irányított esethez tartozó számadatokat nem szerepeltettem a dolgozatomban, ugyanis a NOTEARS algoritmus csupán annyit állít [16], hogy a generált DAG a megfelelő ekvivalencia osztályból kerül ki, melyeknek átfogó vizsgálata meghaladja a dolgozatom kereteit. Annyi azonban megfigyelhető a futtatások során, hogy az olyan V-struktúrákat sem találta el mindig, amelyek léte sejthető a változók jelentése alapján. Ez indokolttá teheti a priori tudás bevitelének lehetőségét a modellbe, ami az eredeti NOTEARS algoritmust használó modellben [16] nem lehetséges. Ennek vizsgálata a későbbiekben érdekes és indokolt lehet, jelen dolgozatban azonban az eredmények egyértelműen azt mutatják, hogy az irányítatlan gráfok összehasonlítása, elemzése javasolt.



4.6. ábra. Az U kódú partner lokális struktúrája

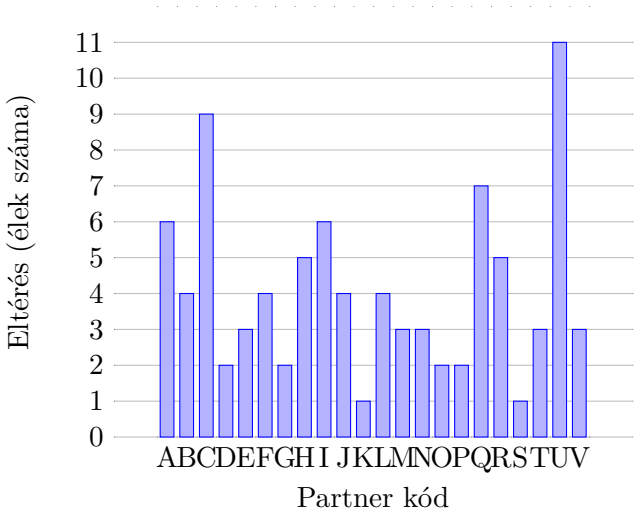


4.7. ábra. A C kódú partner lokális struktúrája

4.1.3. A globális és lokális struktúrák összehasonlítása

Az egyes partnerek struktúráinak a konszenzus-gráftól való különbözőségének vizsgálata után érdemes megnézni azt is, a globális gráftól való eltérés hogyan viszonyul ehhez. A két eltérés abszolút értéke ugyanis egyáltalán nem biztos, hogy egyenlő: az irányítatlan konszenzus-gráf a 4.4-es ábrán látható, míg az ugyanilyen $\lambda = 0.1$ és $\omega = 0.2$ paraméterekkel generált globális gráf a 4.1-as ábrán található. Észrevehető, hogy a globális gráf minden olyan élet tartalmaz, amit a konszenzus-gráf is, de ezen felül még három plusz él is szerepel benne.

A 4.8-es ábrán láthatók a partnerek egyenkénti strukturális Levenshtein-távolságai a globális gráfhoz képest irányítatlan esetekben.



4.8. ábra. A partnerek lokális struktárinak eltérése a globális gráftól irányítatlan esetben.

Az alábbi megállapítások tehetők a grafikon alapján:

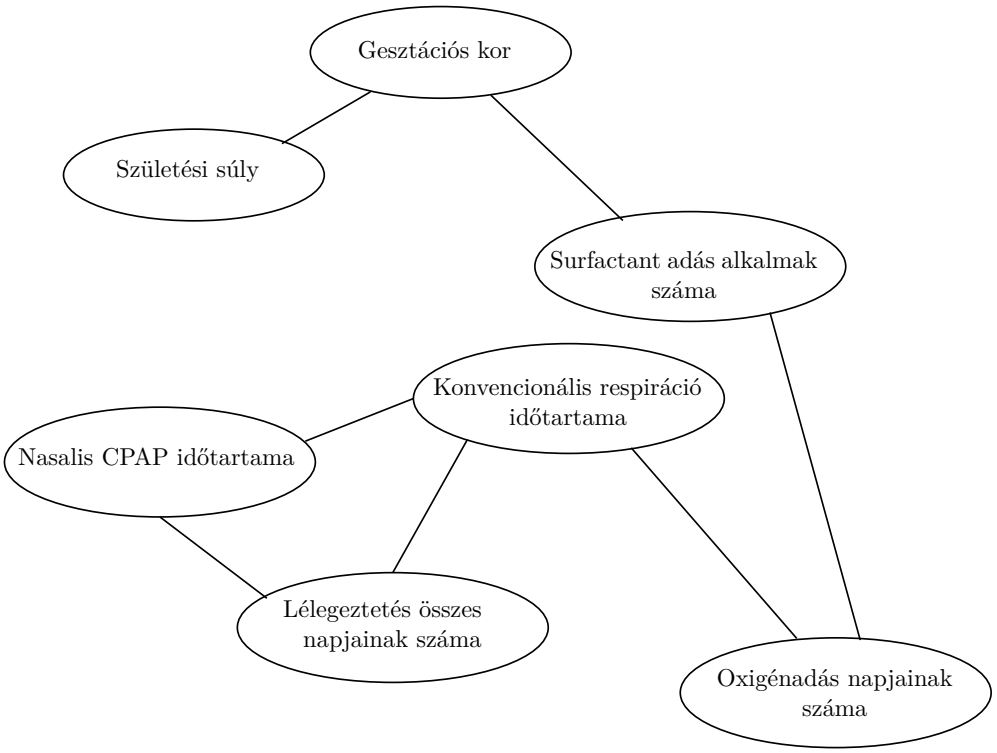
- Az átlagos eltérés irányítatlan élek esetén 4.1 él. Ez több, mint a konszenzus-gráftól számított átlagos eltérések. (Ugyanakkor a globális gráf 3-mal több élet is tartalmaz a konszenzus-gráfhoz képest.)
- A globális gráfhoz legközelebb az S, és a K centrumok állnak, amelyek eltérése mindössze 1 él.
- Legnagyobb távolságra a globális struktúrától az U és a C kódú centrumok állnak. Ugyanaz a két centrum, amelyik a konszenzus-gráftól is legnagyobb távolságra volt. Struktúráik megtalálható a 4.6 és a 4.7 számozású ábrákon.

A globális gráftól való eltérések vizsgálata során mindenképp figyelembe kell venni, hogy itt a nagyobb partnerek sokkal erősebben (akár 20-szor annyi adattal is) reprezentálhatják magukat. A dolgozatomnak nem célja ezeknek a különbségeknek a szakmai elemzése, csupán bemutatom, a strukturális tanulás segítségével hogyan és milyen összefüggések lelhetők fel az adatban.

4.2. Federált tanulás az adatokból

4.2.1. Federált tanulás a teljes adathalmazon

A 3.3.2 fejezetben rögzített paramétereket használva, a teljes adatból a 4.9-es ábrán látható gráf generálható.



4.9. ábra. A NOTEARS-ADMM által federáltan tanult gráf a teljes adatból.

Ez a globális gráffal összevetve, amely a 4.1-es ábrán látható ($\lambda = 0.1$ és $\omega = 0.2$ paraméterezéssel) az irányított és irányítatlan Levenshtein-távolság is éppen 6-ra jön ki. Megfigyelhető viszont, hogy a fontosabb éleket mindkét algoritmus berajzolja.

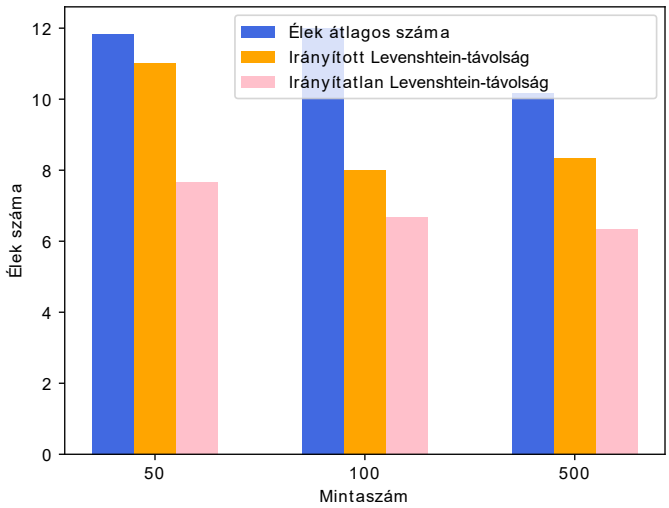
4.2.2. Federált tanulás bootstrap adathalmazokon

A teljes adathalmazból visszatevéses mintavétellel kiválasztott bootstrap [10] adathalmazokon is meg lehet valósítani a tanulást federált és globális esetben is. Ezeknek az eredményeit összehasonlítva számszerűsíthető a két algoritmus által generált gráfok különbözősége. A 4.1-mes táblázatban összefoglaltam a federált algoritmusnak adott méretű (n=330, n=1100, n=1760, n=6600) bootstrap adathalmazokon a teljesítményét, esetenként 10 futást véve figyelembe. (A pontos mintaszámok megválasztása elsőre furcsának tűnhet, de a 4.2-es táblázattal összevetve már érthető lesz.)

Mintaszám	330	1100	1760	6600
Átlagos élszám	10.1	9.6	9.4	10.1
Levenshtein-távolság irányított élek esetén	6.2	4	3.16	7
Levenshtein-távolság irányítatlan élek esetén	5.53	3.78	2.36	5.89

4.1. táblázat. A NOTEARS-ADMM futási eredményeinek különbségei véletlenszerűen választott bootsrtap adathalmaz esetén, megkötés nélkül az egyes partnerek adathalmazainak ezen belüli mintaszámára

A 4.10-es ábrán pedig 3 különböző méretű bootstrap adathalmaz (n=50, n=100 és n=400) esetén 5 futásra átlagolva látható a NOTEARS és a NOTEARS-ADMM által generált gráfok éleinek száma és Levenshtein-távolsága. Leolvasható, hogy meglehetősen nagy különbség van az algoritmusok eredményei között.



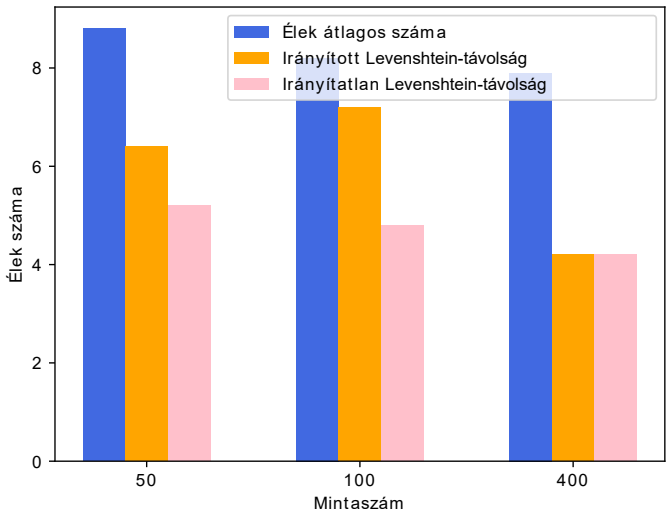
4.10. ábra. n=50, n=100 és n=400 mintaszám esetén a globális és federált algoritmusok által generált gráfok átlagos élszáma és távolságai

A globális adatból federált tanulási protokoll keretében épített gráfban a nagyobb centrumok erősebben, a kisebbek kevésbé erősen képviseltetik magukat. Ez azért van, mert N darab véletlenszerűen kiválasztott minta között várhatóan a nagyobb mintaszámú partnerek adatai gyakrabban fordulnak elő, mint a kisebb partnereké. Ha egy olyan federált gráfot szeretnénk tanulni, amelybe minden partner azonos súllyal számít, lehetőség van minden partner adathalmazából visszatevéses mintavétellel olyan azonos mintaszámú, bootstrap adathalmazok képzésére, melyek a federált tanulást végző algoritmusnak bemenetként adhatók. Így egy olyan gráfot kaphatunk eredményül, amelyben minden partner egyenlő súllyal számít. A 4.2-es táblázatban négy különböző bootstrap adathalmaz mérettel futtattam a NOTEARS-ADMM algoritmust az egyenkénti partnerek esetén. Mivel összesen 22 partner van az adatbázisban, a teljes mintaszám ezeknek az értékeknek a 22-szerese. Az eredményeket minden mintaszám esetén 10 különböző futás átlagából számítottam. Látható, hogy n=15 mintaszámú adathalmazok esetén még nem teljesít igazán jól az algoritmus, n=50 és n=80 esetben már majdnem fele annyit hibázik az n=15 esethez képest. A legjobban n=300 esetén teljesít, amikor átlagosan csak 1 él a különbség egy-egy bootstrap adathalmazból generált gráf között irányítatlan élek esetében.

Partnerenkénti mintaszám	15	50	80	300
Teljes mintaszám	330	1100	1760	6600
Átlagos élszám	10.2	9.7	8.9	9.3
Levenshtein-távolság irányított élek esetén	5.82	3.36	3.8	3.22
Levenshtein-távolság irányítatlan élek esetén	4.84	2.68	2.73	1.16

4.2. táblázat. A NOTEARS-ADMM futási eredményeinek különbségei véletlenszerűen választott bootsrtap adathalmaz esetén, azonos mintaszámú partner adatokkal

A 4.1-es és a 4.2-es diagramokat összevetve kiderül, hogy az algoritmus jobban teljesít abban az esetben, ha az egyes partnerek mintaszámai megegyeznek a teljes mintában. n=80 esetén a legkisebb a különbség, n=15 és n=50 esetén is ez körülbelül 1 él különbséget jelent az irányítatlan éleket tekintve, n=300 esetén viszont már több, mint 5 él a különbség. Igaz, ekkor a rajzolt gráfok átlagos élszáma sem egyezik. Tehát az az észrevétel tehető, hogy a NOTEARS-ADMM algoritmus biztosabb, ha a partnerek adatméretei megegyeznek.



4.11. ábra. Partnerenként n=50, n=100 és n=400 mintaszámú bemenetet fogadó federált és globális algoritmusok által generált gráfok átlagos élszáma és távolságai

A hasonló módon, minden partnertől $n_1 = n_2 = ... = n_k$ mintát a bemenetére fogadó federált algoritmus teljesítménye is összehasonlítható a globális algoritmus teljesítményével.

A 4.10-es és a 4.11-es oszlopdiagramok alapján az értékelhető, hogy a federált és a globális algoritmusok eredménye nagyjából ugyanannyira különbözik egymástól akár azonos partnerenkénti mintaszámot alkalmazunk a NOTEARS-ADMM esetén, akár nem. Valamivel nagyobb a hasonlóság azonos partnerenkénti mintaszám mellett, ebben az esetben az összes mintaszám növelése is javít még a helyzeten.

A federált algoritmusról elmondható, hogy több élben külöbözik az ugyanabból az adathalmazból generált globális gráftól. A hasonlóság mértéke nem mondható nagynak, irányítatlan esetben körülbelül mindegyik második él különbözik. Maga a federált algoritmust bootstrap adathalmazokon vizsgálva azonban szép eredményeket hoz, megfelelően nagy adathalmaz esetén. Ezeken az eredményeken még lehet javítani, ha minden partnertől azonos számú mintán futtatjuk az algoritmust.

5. fejezet

Diszkusszió

Mivel a dolgozatom tárgya elsősorban a NIC adatbázisból való tanulás mérnöki oldala, konklúzióként a munkám mérnöki részeit szeretném kiemelni.

A NIC adatbázis esetén az első megoldandó kérdés a hiányzó adatok kipótlása volt. Az egyes centrumok lokális átlagát és szórását megosztva lehetséges volt az egyes változók globális eloszlása alapján pótolni. A használt változók esetében feltételezhető, hogy normális eloszlásúak vagy ahhoz közel esnek, így a kipótlás után a standardizálásukra is szükség volt. A tapasztalat azt mutatta, ezt a lépést érdemes elvégezni, ugyanis az algoritmusok kimenetei sokkal kiszámíthatóbbak lesznek ilyen módon.

A kipótlás robusztusságát kétféle módon lehetett mérni: egyrészt, a kipótlás előtti és kipótlás utáni korrelációs mátrixok összehasonlításával, másrészt a generált gráfok különbségeinek vizsgálatával. Ezeket a módszereket alkalmazva nagyon biztató eredmények jöttek ki, melyek azt mutatják, jól alkalmazható a NIC adatbázis esetén az eloszlás alapján történő kipótlás. Ki kell azonban emelnem, hogy én aránylag magas kitöltöttségű változókkal dolgoztam, az adatbázisban sok olyan változó is van, amelyek jóval alacsonyabb kitöltöttséggel rendelkeznek. Ezek általában orvosi szempontból is kevésbé releváns adatokat rögzítenek, ha azonban mégis egy ilyen változó vizsgálatára is sor kerülne, a kipótlás során körültekintően kell eljárni.

Az algoritmusok felparaméterezése talán a legtöbb kérdést felvető probléma. Nagyon kevés dologba lehet kapaszkodni a különböző értékek tesztelése során, mivel éles adatokról van szó. Természetesen az algoritmusokat publikáló tanulmányokban vizsgált paraméterértékek támpontot nyújtanak a munka során, azonban ezeknek az értékeknek a további hangolása is mindenképpen szükséges. Amelyik paraméterekről pedig nem szerepel adat az eredeti tanulmányokban, azoknak a beállítása is lehetséges. Ez a gyakorlatban mindenképp nehezebb feladatot jelentett.

Mind a NOTEARS algoritmus hiperparaméter-beállításaira, mind a NOTEARS-ADMM beállításaira tettem javaslatot, ezek a konkrét értékek megtalálhatók a 3.2 és a 3.3 számozású táblázatokban. Egy még pontosabb és jobb paraméterbeállításhoz valószínűleg orvosszakértők nagyobb szintű bevonása lenne szükséges, akik jobban meg tudják ítélni a generált gráfok alapján, melyik egy jó irány, illetve mikor lenne még szükség hangolásra.

Érdekes tapasztalat volt, hogy a NOTEARS-ADMM sokkal érzékenyebb volt a paraméterbeállításokra, mint a NOTEARS algoritmus. Megfigyelhető volt, hogy nem megfelelő beállítások esetén jóval több időre volt szüksége a megoldás megtalálásához, illetve nem megfelelő kimeneteket adott. Vele szemben a NOTEARS a paraméterbeállítások nagy részére formailag helyes eredménnyel futott, különösen nagy időigény növekedés nélkül, viszont egy-egy paraméter módosítása jelentős eltéréseket eredményezett a kimeneti gráfban.

Amikor az eredményekre tekintünk, nagyon fontos, hogy mérnöki szempontból tekintsük őket: bármiféle orvosi következtetés levonásához szakértőkre van szükség és nagyon

alapos vizsgálatok, tesztelések szükségesek előtte. Jelen TDK dolgozatnak ez egyáltalán nem tárgya. Azonban mégis számos tanulság levonható az algoritmusok bemutatott alkalmazásából.

Talán ami leginkább szembetűnő az egész dolgozatból, az az algoritmusok alacsony szintű robusztussága. Mind a NOTEARS, mind a NOTEARS-ADMM meglehetősen különböző eredményeket generált, amikor véletlen mintavételezéssel képzett új adathalmazt kaptak bemenetül. Ezek az eredmények valamelyest javultak a NOTEARS esetében, amikor a partnerek lokális adathalmazain futott az algoritmus. Ez mindenképpen azt igazolja, hogy az adatok mögött rejlő struktúra közel sem egységes a NIC adatbázis esetében. A centrumok oksági struktúrája már robusztusabb, azonban ez is centrumonként eltérő. Lehetséges, hogy még egy további változót, esetleg az adatrekordok rögzítési idejét figyelembe véve, aszerint szeparálva a lokális adathalmazokat egységesebb struktúrákat nyernénk. (Például elképzelhetőnek tartom, hogy valamikor az időben született egy központi, minden centrumra kiterjedő stratégiai döntés, amely befolyásolta egy kezelés alkalmazásának módját/gyakoriságát. Sőt, akár lokálisan az egyes centrumokban is születhettek ilyen döntések, amelyek okozhatnak inkonzisztenciát az adatban.) Mindenképp érdekesnek tartanám az átlagtól nagyobb eltérést mutató centrumok adatainak részletesebb vizsgálatát, bár valószínűleg ehhez további orvosi szakértelemre lenne szükség.

Azt is fontos kiemelni, hogy bár az algoritmus futásának eredménye sok esetben nem volt egységes, az alapvető összefüggéseket szinte kivétel nélkül mindig behúzta a NOTEARS és a NOTEARS-ADMM is. Ez arra enged következtetni, hogy az algoritmusok alapvetően jól működnek, azonban a triviálisakon kívül kevés a nagyon erős összefüggés az adatban. Ezeknek a felismerésén talán még lehet segíteni paraméterhangolással vagy szakértői együttműködés bevonásával, ami segít igazolni, illetve cáfolni egy-egy kérdéses kapcsolat létét.

A dolgozatomban szisztematikusan megvizsgáltam az algoritmusok teljesítményét irányítatlan élek mentén, az irányított élek esetére csak időnként tettem utalást, ugyanis az élek irányának eltalálásában nagyon sokszor tévedett az algoritmus. Azonban ahogyan ezt már kiemeltem, az alkalmazott algoritmusok csak az ekvivalencia-osztály azonosítására alkalmasak, aminek a vizsgálatával a dolgozatom nem foglalkozik. Általában priori tudás bevitelével lehet segíteni a helyzeten, azonban se a NOTEARS, se a NOTEARS-ADMM algoritmus nem alkalmas erre jelen formájában. Lehetséges lenne azonban beépíteni a modellbe egyfajta büntetést és jutalmazást annak függvényében, hogy az algoritmus által megtalálni vélt élek iránya egyezik-e a szakértő által a rendszerbe bevitt élek irányával.

A két algoritmus összehasonlítását tekintve az eredeti NOTEARS az, amelyik stabilabbnak mutatkozott különböző paraméterbeállításokra. Már a paraméterbeállítások során megfigyelhető volt, hogy a NOTEARS-ADMM csak kevés paraméterbeállítás esetén konvergál és ad értelmezhető eredményt. A NOTEARS-ADMM által valósulhat meg ugyanakkor az adatok elosztott kezelése, mely bizonyos szempontból felülmúlja az algoritmus minden hátrányát. A két algoritmus közötti választásnál ezeket a szempontokat mindenképpen mérlegelni kell.

Úgy gondolom, hogy mind a NOTEARS, mind a NOTEARS-ADMM használható a NIC adatbázisból való DAG struktúra tanulására. Az eredményeiket ugyanakkor mindenképp fenntartással kell kezelni, szükség van szakértők bevonására és nagyon sok tesztelésre, hiperparaméter-vizsgálatra. Ezeket szem előtt tartva és sok munka a befektetésével azonban lehetőség nyílhatna a Bayes-hálók gépi tanulásának alkalmazására a magyar egészségügyben. Ez újfajta összefüggések megvilágításával segíthetné az osztályon dolgozók munkáját, ami egészen biztosan mindannyiunk érdeke.

Köszönetnyilvánítás

Szeretnék köszönetet mondani Dr. Antal Péter konzulensemnek, aki a TDK dolgozatom megvalósítása során végig segítette a munkám, és Dr. Szabó Miklósnak, aki az orvosi kérdéseket illetően állt rendelkezésre. Szeretném megköszönni Dr. Valek Andreának is a segítségét, aki szintén az orvosi oldallal kapcsolatos egészen gyakorlati kérdésekben nyújtott segítséget.

A kutatás az Európai Unió támogatásával valósult meg, az RRF-2.3.1-21-2022-00004 azonosítójú, Mesterséges Intelligencia Nemzeti Laboratórium projekt keretében, illetve a TKP2021-EGA-02 számú projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott támogatásával, a TKP2021-EGA pályázati program finanszírozásában.

Irodalomjegyzék

[1] Al-Mohy – Awad H. – Higham – Nicholas J.: A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications,*, 2009.

[2] Antos András – Antal Péter – Hullám Gábor – Millinghoffer András – Hajós Gergely: *Valószínűségi döntéstámogató rendszerek*. 2014, Typotex.

[3] Keith Bonawitz – Hubert Eichner – Wolfgang Grieskamp – Dzmitry Huba – Alex Ingerman – Vladimir Ivanov – Chloe Kiddon – Jakub Konecny – Stefano Mazzocchi – H Brendan McMahan és mások: Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[4] Keith Bonawitz – Vladimir Ivanov – Ben Kreuter – Antonio Marcedone – H Brendan McMahan – Sarvar Patel – Daniel Ramage – Aaron Segal – Karn Seth: Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[5] Stephen Boyd – Neal Parikh – Eric Chu – Borja Peleato – Jonathan Eckstein: *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. 2010, Foundations and Trends in Machine Learning.

[6] H. Bunke: On a relation between graph edit distance and maximum common subgraph. *Elsevier Pattern Recognition Letters*, 1997.

[7] Zhichao Chen – Zhiqiang Ge: Directed acyclic graphs with tears. *IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE*, 2022.

[8] Debo Cheng – Jiuyong Li – Lin Liu – Kui Yu – Thuc Duy Le – Jixue Liu: Discovering ancestral instrumental variables for causal inference from observational data. *arXiv:2206.01931v1*, 2022.

[9] Chickering – David Maxwell: *Learning from Data. Lecture notes in statistics (Capter: Chapter Learning Bayesian networks is NP-complete)*. 1996, Springer.

[10] Bradley Efron – R.J. Tibshirani: *An Introduction to the Bootstrap*. 1994, Chapman and Hall/CRC.

[11] heropup (pseud.): How do i combine standard deviations of two groups? `https://math.stackexchange.com/questions/2971315/how-do-i-combine-standard-deviations-of-two-groups`.

[12] Ignavier Ng – Kun Zhang: Towards federated bayesian network structure learning with continuous optimization. *arXiv:2110.09356v2*, 2022.

[13] White House Office of Science – Technology Policy: Blueprint for an ai bill of rights. `https://www.whitehouse.gov/ostp/ai-bill-of-rights/`.

[14] European Union: Ai act: first regulation on artificial intelligence. https://www.europarl.europa.eu/news/en/headlines/society/20230601ST093804/eu-ai-act-first-regulation-on-artificial-intelligence.

[15] MATTHEW J. VOWELS – NECATI CIHAN CAMGOZ – RICHARD BOWDEN: D'ya like dags? a survey on structure learning and causal discovery. *arXiv:2103.02582*, 2022.

[16] Xun Zheng – Bryon Aragam – Pradeep Ravikumar – Eric P. Xing: Dags with no tears: Continuous optimization for structure learning. *rXiv:1803.01422v2*, 2018.

[17] Kai Zhong – Ian E.H. Yen – Inderjit S. Dhillon – Pradeep Ravikumar: Proximal quasi-newton for computationally intensive l1-regularized m-estimators. *Advances in Neural Information Processing Systems*, 2014.