**Budapesti Műszaki és Gazdaságtudományi Egyetem**
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Kiss Ádám

# INTERPOLÁCIÓALAPÚ IGAZSÁGOSSÁGVIZSGÁLAT BEOSZTÁSOKON

KONZULENS

Jáhn-Erdős Szilvia

BUDAPEST, 2023

# Tartalomjegyzék

# Összefoglaló

Az igazságvizsgálat egy viszonylag fiatal, de annál fontosabb tudományág. Annak egy speciális alfejezete a munkateher-szétosztás, ahol arra kell törekedni, hogy az alkalmazottak minél egyenletesebb munkateher elbírálást kapjanak.

Több különböző módszer és eljárás került kidolgozásra az évtizedek alatt, én egy új, Lipschitz-interpoláción alapuló eljárást dolgoztam ki, hogy kiszámítsam egy-egy alkalmazott munkaterhét. Az eljárás a záróvizsgabeosztás egy egyszerűsített részproblémáján dolgozik, de bővíthető a teljes záróvizsgabeosztás igazságossának eldöntésére is az általános megközelítése miatt. Valódi teszthalmaz részhalmazán lett vizsgálva az eljárás, ami a Budapesti Műszaki és Gazdaságtudományi Egyetem záróvizsgabeosztásán alapult.

Ezt az új megközelítést vetettem össze az irodalomban már ismert módszerekkel. Az eredmények alapján kijelenthető, hogy az interpolációs eljárással igazságos munkateher állapítható meg..

# Abstract

Examining fairness is a relatively new, but indeed very important field of study. A special case of it is the scheduling problem, where the main goal is to assign workers where their workload is balanced and fair.

Many approaches were proposed over the years, and I created a new algorithm based on Lipschitz interpolation to determine a worker's workload. The algorithm works on the subproblem of the original final exam scheduling problem, but it can be expanded to cover the whole problem since it has a general approach. The algorithm was tested on a real dataset, which was based on the Budapest University of Technology and Economics's final exam scheduling.

This new approach was compared with other, already defined methods. The results show that a fair workload can be calculated with the interpolation method.

# 1 Bevezetés

A következő fejezetben először bemutatom az igazságkutatás hátterét, a személyes motivációmat a téma iránt. Ezután kitérek a felhasznált matematikai tételek, eljárások alapjaira, és hogy miért tudtam őket felhasználni kutatásom során.

## 1.1 A probléma leírása

Az igazság egy nehezen megfogható, filozofikus téma. Gimnáziumi éveim alatt számtalanszor éreztem azt, hogy bizonyos, az osztályom hétköznapjait is nagy mértékben befolyásoló döntéseket kifejezetten igazságtalanul hoztunk meg. Ezek között számtalan különféle kérdéskör volt, kezdve a különböző szakkörök időpontjaira, a szalagavatói tánc párjaira, illetve hogy az osztálykirándulás során az osztálypénzzel hogyan bánjunk. Akkor sok döntés igazságtartalmát vitattam, de nem volt alkalmam "fellebbezni". Mesterképzésem során kezdtem többet foglalkozni az igazság metrizálásával, és akkor jöttem rá, hogy az általam tapasztalt problémák nagyobb volumenekben, nagyságrendekben is szereplenek az élet minden területén - mint például az egészségügy, közgazdaságtan, menedzsment.

A legújabb informatikai trendeknek megfelelően a számos különböző mesterséges intelligencia modellre épülő rendszert terveztek. Habár elsőre azt gondolnák - főleg az informatikában kevésbé jártas döntéshozók -, hogy a mesterséges intelligencia képes teljesen, de legalább nagy mértékben objektív döntéseket hozni, ez sajnos nem feltétlenül az igazság. Számtalanszor láthattunk példát arra [1] [2], hogy egy helytelenül betanított MI képes rasszista és szexista döntéseket hozni, ráadásul a betanítás félresiklását észrevenni nehezebb, mint gondolnánk.

Munkám során több igazságdefinícióval találkoztam, amit a különböző területek különböző kérdései, problémái megoldására készítettek, de minimális absztrakció segítségével átültethetőek majdnem bármilyen másik területre. Ilyen nagyobb problémakör volt például az erőforráselosztás, aminek egy speciális részproblémáját, a beosztástervezéseket vizsgáltam. Különböző beosztások rengeteg cég mindennapjainak a része, így kifejezetten fontos, hogy ezeket az alkalmazottak igazságosnak érezzék, és illeszkedjenek a preferenciáikhoz. Már-már szállóige az a versenyszférában, hogy egy alkalmazottat nagyságrendekkel olcsóbb megtartani, mint lecserélni egy újra, emiatt kifejezetten fontos lehet az, hogy az alkalmazottak ne égjenek ki a preferenciáik semmibevétele miatt, és mondjanak fel cégüknél egy ilyen egyszerűen orvosolható dolog miatt. Ráadásul beosztások igazságtalanságát nehéz észrevenni - mind az alkalmazottaknak, mind az akár szántszándékon kívül igazságtalanságot elkövető döntéshozók számára - így mindenki számára fontos, hogy legyen egy transzparens és az alkalmazottak szemszögéből szimmetrikus metrika, ami segít észrevenni az esetleges problémákat, illetve segíthet a munkateher megállapításában.

A dolgozatom további részeiben egy általam kidolgozott rendszert fogok bemutatni. A bevezetés hátralévő részében a kutatásom matematikai alapjaira térek ki. A második fejezetben összevetem az elméleti alapjait pár ismertebb igazságdefinícióval, azok felhasználásaival. A harmadik fejezetben ismertetem kutatásom menetét, miképp formálódott a végső gondolat, és hogy a felmerülő különféle problémákra milyen megoldások állnak rendelkezésre. A negyedik fejezetben az eredményeim fogom összevetni egy beosztástervező algoritmus kimenetelével.

## 1.2 Matematikai képlet az igazsághoz: a Lipschitz-tulajdonság

Kutatásom során záróvizsgákat vizsgáltam a munkateher szemszögéből. Ezek a beosztások komplexebbek, mint a hagyományos munkabeosztások a különböző megkötések és szerepkörök miatt. Ez a bonyolultsági faktor később meg is jelenik majd a munkateher megállapításának kérdésénél. Amikor igazságos munkaelosztásról beszélünk, azt szeretnénk, hogy hasonló bemeneti tulajdonságú emberek hasonló elbánásban részesüljenek. Ezt a gondolatot nehéz formalizálni, és több próbálkozás is történt rá. Én legjobb formális megfogalmazásnak a függvények Lipschitz-tulajdonságát találtam. [3].

A Lipschitz-tulajdonság [4] matematikai képlete a következő:

$$\|f(x) - f(y)\| \leq M * \|x - y\|$$

ahol $x$ és $y$ f függvény értelmezési tartományának két pontja, $M$ pedig egy maximális megengedett meredekségi érték. A tulajdonság következtében közeli pontok kimenetének bizonyos meredekségen belül kell lenniük, tehát nem történhet meg, hogy hasonló tulajdonságú elemek sokkal eltérő bánásmódban részesüljenek, de képesek vagyunk diverzifikálni.

Innen adódott a gondolat, hogy a munkateher kiszámítása legyen egy Lipschitz-tulajdonságú függvény, ahol a bemenet egy-egy alkalmazott. Az alkalmazottakat ábrázolhatjuk egy m-dimenziós pontként, ahol minden dimenzió egy-egy tulajdonságának felel meg. Ez a tulajdonságmegválasztás önmagában is egy komplex probléma, amire kiváló példa a Német Hitelbank esete [5]. Záróvizsgabeosztások esetén ilyen tulajdonság lehet a tanított tárgyak száma, a kurzusokon részt vevő összes hallgató, a kurzusok átlagos hallgatószáma, vagy akár a betölthető szerepkörök száma. Ezután szükség volt egy olyan függvényre, aminek a kimenete lehet a munkateher, és Lipschitz-tulajdonságú. Mivel egy ilyen függvényt triviálisan találni lehetetlen, így a problémát több részproblémára hasítottam. Az első részprobléma az volt, hogy képes vagyok-e megállapítani egy alkalmazott munkaterhét az összes többi alkalmazott munkaterhének ismeretében. Itt az eredeti probléma valójában át is alakult egy interpolációs problémává, és már nem volt szükségem az "eredeti" függvényre. Ezután már csak egy olyan interpolációs eljárást kellett találnom, ami megtartja a függvények Lipschitz-tulajdonságát.

## 1.3 Lipschitz-interpoláció

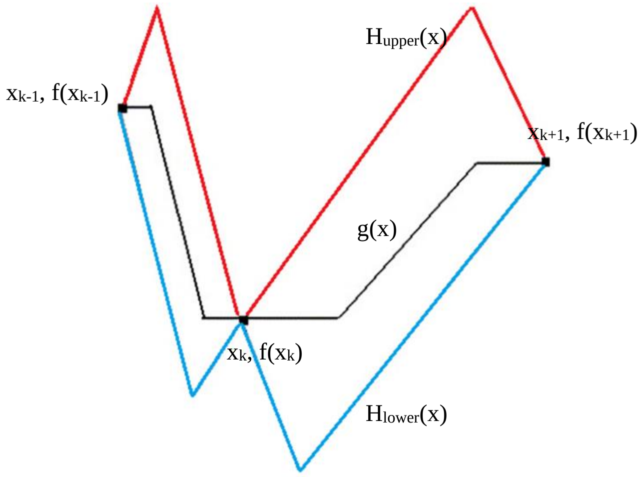
### 1.3.1 Lipschitz-interpoláció tulajdonságai, megkötései és előnyei

Az interpoláció egy matematikai eljárás, aminek lényege, hogy mindössze a függvényértékekből képesek vagyunk nagyon pontos becsléseket végezni a környező pontokra. Így tehát nem szükséges ismernünk magát az egzakt függvényt – sőt, általában még a nagyságrendjét sem – mindössze a függvény által produkált kimeneti pontokra van szükség további függvényértékek nagyon közeli becsléséhez. Ráadásul a legtöbb interpolációs eljárás polinomiális időben kiszámítható, így egy viszonylag olcsó számításról van szó. Viszont fontos megjegyezni, hogy minél jobban eltávolodunk a megadott pontoktól, amik alapján az interpolációt végezzük, annál nagyobb lesz a hiba – sőt, az is előállhat, hogy az eredeti függvény monoton csökkenő azon az intervallumon már, míg az interpolált monoton növekvő. Éppen ezért interpolációt csak bizonyos pontok közelében lehet értelmesen alkalmazni.

Az interpolációs eljárások nagy előnye a transzparencia, így nemcsak a kimenetet tudjuk, hanem a számítás teljes menetét, így a kimenet "miértjeit" is. Másik előnye az interpolációs eljárásoknak, hogy tetszőlegesen sok dimenzióra működnek általában, így az eljárás könnyen átültethető a záróvizsgabeosztások problémájáról sok másik, hasonló jellegű erőforráselosztási problémára. Fontos még, hogy a matematikai jellegükből adódóan az interpolációs eljárások szimmetrikusak.

Szerencsére a matematikában az elmúlt években megnyílt az igény arra, hogy speciális tulajdonságú függvényeket interpoláljunk úgy, hogy az eljárás megőrizze ezeket a speciális tulajdonságait is az eredeti függvénynek. Gleb Beliakov kidolgozott egy olyan interpolációs eljárást[6] ami képes függvények Lipschitz-tulajdonságát megtartani - a Lipschitz-interpolációt. Ennek az eljárásnak viszont van pár bemeneti megszorítása, hogy az interpoláció helyes értéket adjon. Mivel itt egy matematikai képletről beszélünk, az interpolációt akkor is el tudjuk végezni, ha ezek az előfeltételek nem teljesülnek, viszont ekkor - értelemszerűen - hibás eredményeket kapunk. Fontos, és egyáltalán nem triviális feltétel az, hogy a pontjaink egy zárt intervallumból/térből származzanak. Ennél logikusabb megszorítás az, hogy a bemeneti pontoknak egy Lipschitz-tulajdonságú függvény kimeneteinek kell lenniük, tehát a pontoknak lipschitz-eseknek kell lenniük.

Az interpoláció egyik fontos tulajdonsága, hogy a bemeneti pontokban megtartja az eredeti függvényértéket, a többi pont értékének megállapításához pedig felhasznál egy alsó- és felső becslést, és ezek matematikai átlagát vesszük.




**1. ábra A Lipschitz-interpoláció alsó és felső becslései az k-1 és k+1. pontok között. Az interpolált függvény g(x), $H_{upper}$ a felső, $H_{lower}$ pedig az alsó becslés függvénye, Gleb Beliakov ábrázolásában [6].**

## 1.3.2 Az interpoláció menete

Ahhoz, hogy ez az átlag működhessen és számítsanak a pontok távolságai, a becslés felhasznál egy speciális távolsági metrikát. Ez a simplified distance, jele $d_P$. Egy feltevés volt, hogy az interpoláció m-dimenziós pontjai egy zárt térből származzanak, legyen az a tér P. Legyen m darab kanonikus vektorunk, amik rendre:

*h₁* = (-*v*₁, 0, 0, … 0), *h*₂ = (0, -*v*₂, 0, … 0) és *h_m* = (0, … -*v_m*) alakúak. Vegyünk fel egy speciális $h_{m+1} = (-v_{m+1}, -v_{m+1}, \ldots -v_{m+1})$ alakú vektort is, ez az $m+1$ darab vektor által

$$d_P(x, y) = \max\left\{\max_{i=1,\ldots,m} v_i(y_i - x_i),\, v_{m+1}\sum_{i=1}^{m}(x_i - y_i)\right\}$$

**2. ábra simplified distance metrika Gleb Beliakov [6] felírásában, még a kétszeres maximumszámítással**

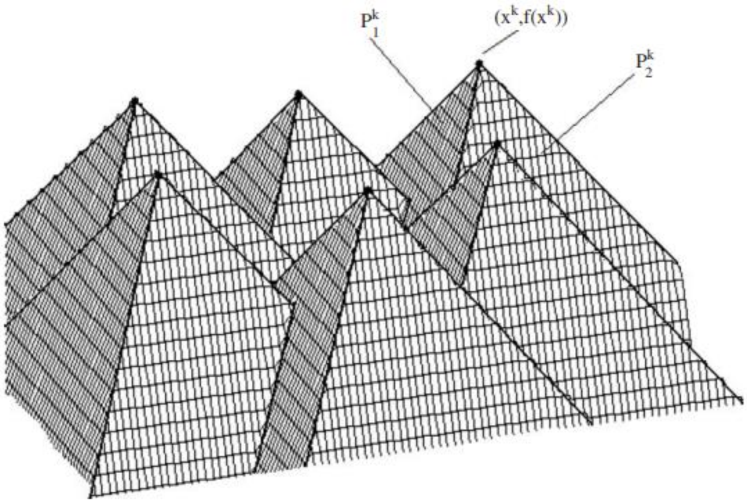
bezárt simplex lesz P, amire értelmezzük a simplified distance metrikát.

Vezessünk be egy slack változót, aminek értékét fejezzük ki *m* segítségével! A slack változó értéke legyen $x_{m+1} + 1 = 1 - \sum_{i=1}^{m} x_i$ ! Ennek a slack változó felhasználásával egy átrendezés után ki tudjuk küszöbölni a kétszeres maximum számítást, és az alábbi képletet kapjuk, ahol ráadásul az *m+1.* vektorunk nem képez kivételt a többi m kanonikus vektorhoz képest: $d_P = \max_{i=1\ldots m+1} v_i * (y_i - x_i)$ Az egyszerűség kedvéért a képletben szereplő minden $v_i$-t 1-nek választottam Ez egy jó választás, hiszen ugyanúgy előáll a zárt *P* simplex, ráadásul a számítások is jelentősen egyszerűsödnek.

A két becslésünk egyenletei pedig a következőek lesznek [6]:

$$H^{upper}(x) = \min_k(y^k + Md_P(x, x^k)$$

$$H^{lower}(x) = \max_k(y^k - Md_P(x, x^k)$$




**3. ábra az interpoláció alsó becslése két dimenzióban, Gleb Beliakov ábrázolásában [6]**

Vegyük észre, hogy a Lipschitz-interpoláció képletében a felső becslés kifejezhető az alsó becslésből átrendezésekkel. Emiatt az interpoláció becsléseinek oldalai páronként párhuzamosak, meredekségeik egymás ellentettjei. Emellett vegyük észre, hogy mindkét becslés értéke valójában 1-1 pont értékétől függ. Ez egy nagyon fontos tény, amikor azt keressük, hogy a kapott interpolált érték "miért" annyi, melyik (legfeljebb) 2 pont miatt kapta az értékét.

# 2 Az igazságvizsgálat kutatása

Ebben a fejezetben meg fogom vizsgálni az igazságkutatás irodalmi hátterét. Bemutatok pár ismertebb, elfogadott igazságdefiníciót, és azok felhasználási területeit. Ezek után összevetem az elméleti háttereiket, alapjaikat az én módszerem elviségével.

## 2.1 Az igazságvizsgálat eredete

Az igazság kutatása az 1960-as évektől kezdve egyre fontosabb terület lett. Nagyon széles spektrumot fed le, kezdve repülőgépek késleltetéséről [7], szervdonorokon [8] [9], hálózatirányításon [10], közgazdaságtanon [11] át, egészen a hitelbírálatig [5].

Sokáig az volt a fő irányvonal, hogy minimalizáljuk az összköltséget - ahol az összköltség lehet az összes repülőgép várakozásának ideje, az alkalmazottak megszegett preferenciáinak száma, hálózaton kézbesítetlen csomagok száma - viszont ez az elmúlt években változott. A "hagyományos" megközelítés inkább volt egy matematikai optimalizációs probléma, és próbálták meg a való élet problémáit megoldani egy már jól ismert és kutatott matematikai problémára visszavezetni, amire hatékony algoritmusok armadája áll rendelkezésünkre. Ilyenek például hagyományosan a maximális párosítás [12], lineáris programozás [13]. Ezek a megközelítések valóban minimális összköltségű megoldásokat produkáltak, de ennek érdekében akár "feláldoztak" bizonyos egyéneket. A beosztástervezések problémájánál egy klasszikus példa az, hogy egy kis létszámú cégnél minden alkalmazott egy kényelmes, preferenciáihoz illeszkedő beosztást kapjon - egyik kivételével, akinek a beosztása úgy épül fel, hogy a senki által sem preferált időpontokba osztják be, nem törődve a puha megkötésekkel, mint például kényelmes ebédszünet.

Pont emiatt sokkal inkább előtérbe került az úgynevezett "max-min fairness" [7]. Ahogy neve is mutatja, a hangsúly áthelyeződik arra, hogy minimalizáljuk a maximális költséget. Tehát itt már nem az összköltségről van szó, hanem arról, hogy egyik egyén se kapjon kiugróan rossz elbánást a többihez képest. Matematikai értelemben véve az ilyen megoldások szinte esélytelenek, hogy optimálisak legyenek a hagyományos értelemben, mégis sokkal igazságosabb eredményekhez vezetnek.

## 2.2 Modern igazságdefiníciók

Az irodalomban leggyakrabban öt igazságaxiómát [7] használnak fel, ezek a közgazdaságtanból erednek, de általánosan is felhasználhatóak, minimális módosítással kiterjeszthetőek különböző problémakörökre is.

I. Axióma: Pareto-optimalitás: az elosztás során nem keletkezik veszteség, az erőforrás teljes mértékben szétosztásra kerül

II. Axióma: Szimmetria: két azonos elem elbánása egyenlő lesz

III. Axióma: Affin invariancia: tranzformációfüggetlenség, az eloszlás nem változik meg más referenciaértékek használatában (ez utóbbi közgazdaságtani definíció)

IV. Axióma: Függetlenség az irreleváns alternatívákra nézve: *A* elem kiválasztása *B* helyett nem függ a többi elemtől.

V. Axióma: Monotonitás: Egyik elem elbírálásának változtatása a másik elem elbírálásának változásával jár.

Az axiómák érdekessége, hogy nincs olyan eljárás, ami mind az öt axiómát teljesítené. Nash [8] eljárása az első négy axiómát teljesítette, de idővel megjelentek alternatívák [9] is amik különböző axiómákon [10] alapultak. Az én eljárásom a 2-5 axiómák szerint készült, a szimmetriáért feláldoztam a paretooptimumot.

### 2.2.1 Összevetés a Max-min fairness-el

A max-min fairness egyik hagyományos módszere a Kalai-Smorodinsky módszer [10]. A módszer az első három és az ötödik axiómát elégíti ki, pár megszorítással. Az egyik ilyen kikötés az, hogy két játékos szerepel, máskülönben az első axióma, vagyis a paretooptimalitás,– amit az én módszerem sem tud teljesíteni – néha csorbul.

Ez a módszer azzal operál, hogy a lehető legkisebb legyen a legnagyobb veszteség, míg az én megközelítésem szerint hasonló egyedeknek hasonló elbánásuknak kell lenniük. Emiatt nálam nem prioritás a legnagyobb veszteség minimalizálása, sokkal inkább annak a „jutányossága”, arányossága a fontos.

A max-min fairness főleg és elsősorban hálózati kommunikációs problémákra alkalmazzák – amikre példát láthattunk Huang és Bensaou [11] esetében, Hahne összevetésében [12], vagy Sridharan és Krishnamachari eljárásában [13] – , amire igazságosabb megoldást eredményez, mint az én, eredetileg beosztásokra tervezett megoldásom átültetése.

### 2.2.2 Összevetés Nash módszerével

Nash módszere egy hagyományosabb megközelítés, mint Kalai-éké, és a közgazdaságtanból ered, de szintén sokszor használják hálózati kommunikációs problémákon. Nash módszere általánosabb, mint az előbb említett eljárás, és habár az eredeti verzióban ott is két szereplő kapcsolatáról írtak, ennek van általánosított változata több szerepőre is [14].

Nash módszerének a lényege, hogy egy nem kooperatív helyzetben két szereplő (akik ismerik egymás stratégiáit) egyensúlyi pozícióban vannak, ha a szereplők ismerik egymás stratégiáikat, és azon változtatva nem tudnak nagyobb nyereséget szerezni. Ennek a közgazdaságtani definíciónak az igazságvizsgálatban proportional fairness néven létezik alkalmazása, amit szintén gyakran hálózatok esetében alkalmaznak, mint például Jiang és Liew lokális és ad hoc hálózatokra kidolgozott módszere [15], vagy Li, Pal és Yang sávszélességelosztásra kidolgozott eljárása [16].

A módszer tehát igazságosnak egy elosztást a többi lehetséges elosztáshoz képest viszonyítva nevez. Ezzel szemben az én megközelítésemben nem vizsgálok meg különböző lehetséges elosztásokat ahhoz, hogy egyet igazságosnak nevezzek, hanem szigorúan csak az elosztás szereplőinek egymáshoz viszonyulását nézem meg.

# 3 Kutatásom menete

Ebben a fejezetben bemutatom kutatásom folyamatát, kitérve a problémákra, amikbe ütköztem az ötlet megszületésétől a megvalósításig, és azok megoldásaira. Ismertetem a bemeneti adathalmazt, és annak feldolgozását. Végül bemutatom az interpoláción alapuló eljárásom lépéseit, amiket eljárásom felhasznál.

## 3.1 A bemenet, adathalmaz

A bemenet valódi záróvizsgák leszűkítései voltak – eltekintettem a párhuzamosan folyó vizsgák lehetőségétől, valamint az MSc és BSc vizsgák eltérő hosszát sem vettem figyelembe. Hagyományosan egy záróvizsgán több speciális szerepet betöltő tanár szerepel. A vizsgázó diákhoz tartozik a konzulense, akinek személye előre meghatározott, illetve ritkább esetben (speciális, külső témák esetében) egy külső tag, aki nem áll közvetlen jogviszonyban az egyetemmel. Minden vizsgához tartozik egy elnök, titkár és belső tag, akiknek különböző követelményeknek, hard constraint-oknak kell megfelelniük a pozíció betöltéséhez. Pár szerepkör összevonható, tehát egy vizsga esetében egy személy tölti be egyszerre a két tisztséget. A konzulens lehet egyszerre titkár, vizsgáztató, de elnök is akár, viszont az elnöknek és a titkárnak a személyének szigorúan különbözőnek kell lennie miden esetben.

48 tanár és 100 diákot tartalmazott egy xlsx adatfile, amin a munkaterhet vizsgáltam. Több különféle mérőszám lett felhasználva, melyek egy része egy szumma, másik része pedig valamilyen átlag. Figyelembe vettem, hány tárgynál tölthet be vizsgáztató szerepkört az egyén. Szempont volt, hogy átlagosan hány oktató van a tárgyaknál, ahol tanít, és ezek alapján átlagosan hány hallgató jutna az adott oktatóra. Ezen felül szempont volt még az is, hányan hallgatták összesen a tárgyait. Fontos megjegyezni, hogy a konzultált hallgatók számát, mint paramétert nem használtam fel.

Habár a záróvizsga során egy tanár több különböző, nem egyenlően megterhelő szerepkört is betölthet, én végül a vizsgáztatóként, elnökként, titkárként és bizottsági tagként eltöltött órák számát neveztem munkatehernek. Ezen kívül a teljesen nulla értékkel rendelkező elemeket kivettem, mivel a modellem nem támogatja azoknak a használatát, ennek okát később még taglalom.

Eredményeim 1011 bemeneti adatfile-on alapulnak, amik a már említett módon épültek fel. Ezekről részletesebben az Eredményeim fejezetben fogok foglalkozni.

## 3.2 Egy személy vizsgálata egy beosztásban

Mint már korábban említettem, Lipschitz-interpolációt használtam fel, mivel ez az eljárás megőrzi egy – megszorításokkal rendelkező – függvény Lipschitz-tulajdonságát. Minden személy egy $m$ dimenziós pont volt, a dimenziói pedig egy valamilyen tulajdonsága, jelen esetben az előzőleg említett oktatói tulajdonságok.

Kutatásom során megvizsgáltam több lehetséges szempontot arra, hogy egy oktatót metrizálhassak. Sok olyan tulajdonság van, ami nehezen számszerűsíthető. A logikai értékekhez kézenfekvő volt a $\{0,1\}$ halmaz elemeit választani, habár ez nem mindig volt jó megoldás. Túl nagy értékek esetén ez az extra dimenzió nem igazán számított, így ilyenkor arányosan nagyobbítani kell ezt az értéket. Záróvizsgabeosztások kezelése során ilyen eshetőség nem állt fent.

Másik ilyen probléma volt a felsorolásos értékek kezelése. Erre az volt a gondolatom, hogy annyi új dimenziót vegyek fel, ahány lehetséges felsorolási érték szerepelhet, és mindegyik érték legyen 0, kivéve azt a dimenziót, amelyiket megfeleltettünk az adott értéknek. Ekkor is szükséges lehet bizonyos esetben valamilyen fokú arányosítás, de záróvizsgabeosztások esetén ez sem állt fent.

## 3.3 A Lipschitz-interpoláció használatának problémái

### 3.3.1 A bemeneti adatpontok származásának problémája

Több problémába ütköztem először a Lipschitz-interpoláció használata során, amik mind a szigorú megkötésekből származtak. Ezek a megkötések azért kellettek, hogy az interpoláció elvégzése után is Lipschitz-tulajdonságú legyen a kimenet. Az egyik megkötés az volt, hogy az adatok egy lipschitz-es függvényből származzanak, igazságtalan beosztás esetén viszont ez értelemszerűen nincs így. Igazságtalan beosztások esetén ugyanis hasonló tulajdonságú alkalmazottak elbánása nagyságrendekkel eltér, tehát a függvény, ami a munkaterhüket állapítja meg, nem lehet Lipschitz-tulajdonságú. Másik szemszögből: nem Lipschitz-tulajdonságú függvényt nem lehet interpolációval úgy „előállítani”, hogy lipschitz-es legyen.

Ennek a megkötésnek a figyelmen kívül hagyása anomáliákhoz vezetett az interpoláció elvégzése során, mint például az alsó becslés nagyobb értéket vett fel, mint a felső, ez pedig szemlátomást ellentmondás. Szerencsére ezeket az anomáliákat nemcsak áthidalni sikerült, hanem előnyömre fordíthattam amikor igazságtalan elbírálást kerestem annak segítségével, hogy az interpoláció működésének jellegéből eredően képes voltam megválaszolni, melyik adatpontokon alapult ez a helytelen számítás, és emiatt melyik adatpontok között van potenciálisan igazságtalan elbánás.

### 3.3.2 Megoldás az adatpontok származásának problémájára

Egy polinomiális algoritmust fejlesztettem ki a probléma megoldására, melynek lényege pontelhagyáson alapult. Az algoritmus végigiterál minden egyes adatponton, és elhagyja azt a ponthalmazból, minden alkalommal pontosan egyet – tehát a korábban elhagyott pontokat **változtatás nélkül** visszahelyezi mindig az algoritmus.

```
FindUnfairs(allTeachers: Teacher[], outerElements: Teacher[])
        unfairs := []

        for currentTeacher : allTeachers
        {
                if currentTeacher not in outerElements
                {
                        otherTeachers = filter(allTeachers, (teacher: Teacher) => { return teacher.name != currentTeacher.name })
                        lowerBound = Hlower(currentTeacher, otherTeachers)
                        higherBound = Hupper(currentTeacher, otherTeachers)
                        expectedWork = (lowerBound.workload + higherBound.workload)* 0.5

                        if (floor(expectedWork) - margin > original_workload(teacher)) or (ceil(expectedWork) + margin < original_workload(teacher))
                        {
                                hits.add(teacher.name, lowerBound.name, higherBound.name)
                        }
                }
        }
        return unfairs
```

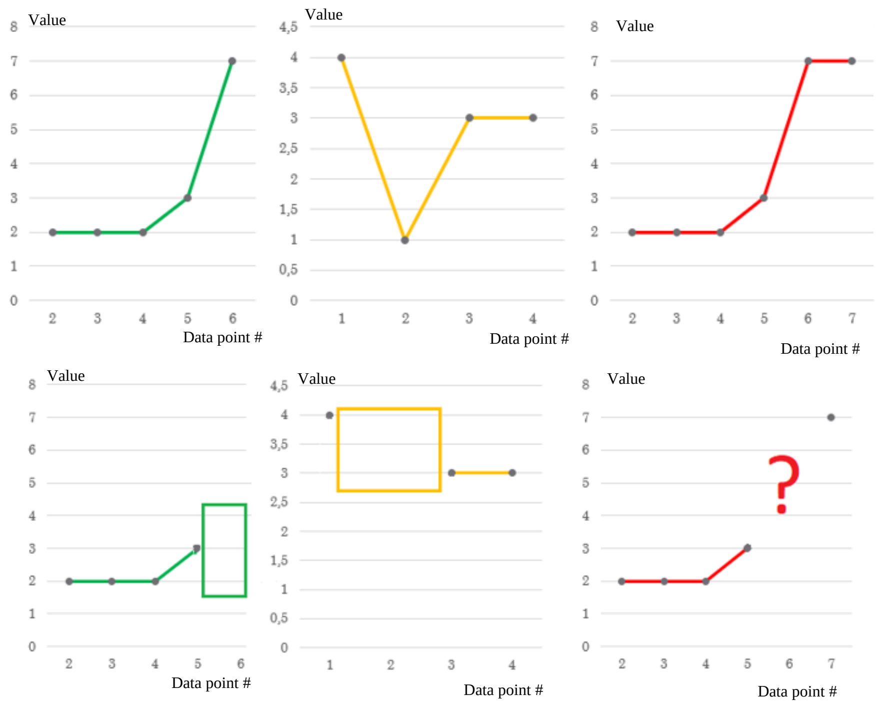
**4. ábra Az interpoláció pszeudokódja. Az outerElements az intervallumok szélén való elhelyezkedés problémája miatt kell, ezt a 3.3.3. fejezetben taglalom részletesebben.**

Az elhagyott pontot ezután megpróbálja az algoritmus „megsaccolni”, elvégezni rá az interpolációt. Amennyiben az eredmény – egy bizonyos hibahatáron belül – megegyezik az eredetivel, akkor igazságos az elbánás. Ellenkező esetben pontosan megállapítható, hogy melyik (legfeljebb) két pontból származik az interpolált érték, így visszakereshető, hogy melyik (legfeljebb) három adatpontot kell megvizsgálni manuálisan. Az algoritmus ilyen mértékű transzparenciája a való életben nagyon előnyös lehet azzal szemben, minthogy annyi a végeredmény, hogy a beosztás igazságos-e vagy sem, de még annál is jóval több információval szolgál, minthogy ha csak annyit állapítana meg, melyik konkrét alkalmazott elbánása igazságtalan.

Az algoritmus pszeudokódját a 4. ábra mutatja be. A Hlower és Hupper a két interpolációs becslőfüggvény változata. Optimalizációs szempontból nemcsak a becsült értéket adja vissza, hanem azt a pontot is, ami alapján a becslést elvégezte – jelen esetben egy tanárt. A függvény kimenete, az unfairs egy rendezett hármas, aminek első eleme az igazságtalan elbánt elem, a második és harmadik pedig azok az elemek, amik alapján az interpoláció készült.

### 3.3.3 Zárt intervallumon való értelmezés problémája

A Lipschitz-interpoláció másik bemeneti kritériuma a zárt téren való értelmezés problémája, ugyanis az interpoláció az utolsó pontokon túl 0 meredekséggel rendelkezik. Ennek oka az, hogy az alsó és felső becslések függvényének adott pontbéli meredeksége megegyezik, és a szélső pontokon túl mindig ugyanazokat a pontokat fogja felhasználni, így a két becslés azonos arányban csökken illetve nő.




**5. ábra A pontelhagyás szemléltetése, nem Lipschitz-tulajdonságú függvényeken**

Emiatt ha szélről hagyunk el pontot, akkor mindenképp rossz interpolált eredményt fogunk kapni, hiszen ekkor új széle lesz az interpolációnak, és azt az értéket fogja a pontunkra vetíteni, ezáltal esélytelen, hogy helyes eredményt kapjunk arra az adott becslésre.

A pontelhagyás menetével az 5. ábra foglalkozik. A zöld függvény esetében az utolsó, a sárga függvény esetében a második pontot hagyja el az eljárás. Becslés szerint a téglalapban veheti fel a lehetséges értékeit, viszont ettől kívül esnek, így megállapítható azt is, hogy a függvény nem Lipschitz-tulajdonságú. A piros függvény esetében az utolsó előtti pont elhagyása van soron, és az említett becslési anomália áll fent, ami miatt itt is észrevehető, hogy a függvény nem lipschitzes.

### 3.3.4 A zárt intervallumon való értelmezés problémájának megoldása

A legkézenfekvőbb és legegyszerűbben megvalósítható megoldás az, ha nem hagyunk el iterációnk során olyan pontokat, amik valamelyik dimenzió legszélén helyezkednek el. Ez a kézenfekvő megoldás viszont azt a veszélyt rejti magában, hogy ezáltal nem iterálunk végig minden ponton, így egy esetleges rossz elbánást nem fogunk kiszűrni. Éppen ezért ki kell küszöbölnünk, hogy ez megtörténhessen.

Szerencsére nincs szükség semmilyen extra lépésre vagy változtatásra, ugyanis ha a szélen lévő érték olyannyira kiugró lenne, akkor a többi pont becslését torzítaná és anomáliát idézne elő, vagy egyszerűen csak a már említett hibahatáron kívülre interpolálná a pontot.

Ráadásul mivel nemcsak a helytelen értéket kapó pontot tudjuk, hanem azt is, hogy melyik pontoktól származik ez az érték, így bármilyen extra lépés nélkül is tudjuk azt, hogy melyik pont miatt sérülhetett a Lipschitz-interpoláció feltétele.

## 3.4 Az összes tanár munkaterhének megállapítása

Az előbb bemutatott eljárás képes volt egy tanár munkaterhét megállapitani. Adódik a felvetés, hogy ennek a módosításával kiszámítható-e az összes alkalmazott munkaterhe anélkül, hogy bármelyiket is előre ismernénk. A gondolat onnan támadt, hogy egy elhagyott pontot képes vagyok pótolni. Ezután a kérdés az, hogy kettő, három, m elhagyott pont esetében is képesek vagyunk-e.

Mindenféle módosítás nélkül ez nem lehetséges sajnos, hiszen az interpoláció csak zárt térben hajtható végre, és egy idő után egy újabb pont elhagyása csökkenti a ponthalmaz geometriai méretét, ami ahhoz is vezet, hogy a csökkentett méreten túlra visszakerülő pontok interpolált értékei helytelenek lesznek. Emiatt több gondolat mentén indultam

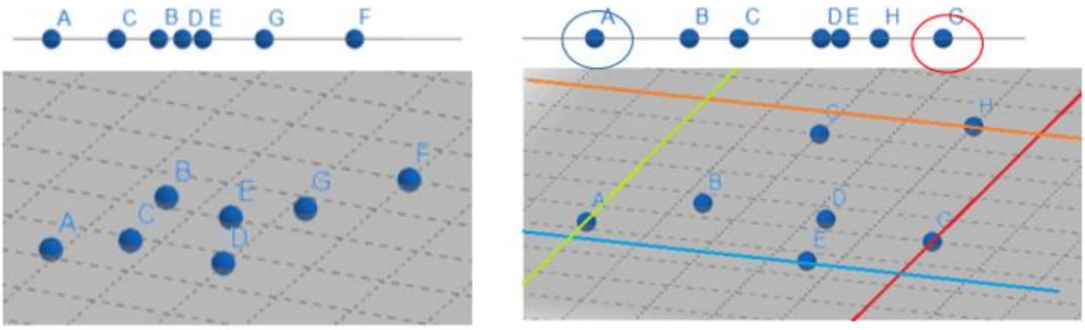
```
Populating with some average workers:
     Name: Artur
     Students: 20     No. subjects: 2     Workload: 21
     Name: Bela
     Students: 15     No. subjects: 1     Workload: 18
     Name: Cintia
     Students: 24     No. subjects: 2     Workload: 22
Teachers:
['Artur', 'Bela', 'Cintia']
Hits:
[]

##########################
 Adding a highly underused employee:

     Name: Naplopo
     Students: 20     No. subjects: 2     Workload: 1
Teachers:
['Artur', 'Bela', 'Cintia', 'Naplopo']
Hits:
[('Artur', 'Naplopo', 'Cintia'), ('Bela', 'Naplopo', 'Artur'), ('Cintia', 'Naplopo', 'Artur'), ('Naplopo', 'Artur', 'Artur')]

##########################
 Adding a highly overused employee:
     Name: Sztahanov
     Students: 2     No. subjects: 1     Workload: 100
Teachers:
['Artur', 'Bela', 'Cintia', 'Sztahanov']
Hits:
[('Artur', 'Bela', 'Sztahanov'), ('Bela', 'Artur', 'Sztahanov'), ('Cintia', 'Artur', 'Sztahanov'), ('Sztahanov', 'Bela', 'Bela')]

##########################
  Adding both of the previous two:
Teachers:
['Artur', 'Bela', 'Cintia', 'Naplopo', 'Sztahanov']
Hits:
[('Artur', 'Naplopo', 'Sztahanov'), ('Bela', 'Naplopo', 'Sztahanov'), ('Cintia', 'Naplopo', 'Sztahanov'), ('Naplopo', 'Artur', 'Sztahanov'), ('Sztahanov', 'Naplopo', 'Bela')]

##########################
   Adding one with lower and one with higher workload:

     Name: Denes
     Students: 23     No. subjects: 3     Workload: 25
     Name: Emese
     Students: 9     No. subjects: 1     Workload: 8
Teachers:
['Artur', 'Bela', 'Cintia', 'Denes', 'Emese']
Hits:
[('Emese', 'Bela', 'Bela')]

##########################
   Adding two more average employee:
     Name: Fulop
     Students: 12     No. subjects: 1     Workload: 13
     Name: Gerda
     Students: 10     No. subjects: 1     Workload: 10
Teachers:
['Artur', 'Bela', 'Cintia', 'Denes', 'Emese', 'Fulop', 'Gerda']
Hits:
[]

##########################
```

**6. ábra Ütközőhármasok szemléltetése egy fiktív adathalmazon.**

Észrevehető, hogy a módszer több hibás pont esetén (legyenek a hibák túl- vagy alulfoglalkoztatásból eredőek vegyesen) mindegyiket észreveszi, valamint újabb pontok képesek lehetnek korábbi hibák korrigálására. Ezzel részletesebben a 6. ábra foglalkozik, ahol bemutatom az ütközőhármasok detektálását.

## 3.4.1 Egyenletrendszerek, korlátok használata

Az egyik első gondolat az volt, hogy minden pontra állapítsunk meg különböző megszorításokat, feltételeket, korlátokat, amik a többi pont matematikai értelemben vett helyéből és értékéből fakad. Ezután folyamatosan egyre több, minél szűkebb feltételt szabok meg, arányosítok, és amikor valamelyik pont konkrét értéket vesz fel, akkor behelyettesítek.




**7. ábra Korlátok használatának problémája. Habár látjuk, hogy egy dimenzós terek esetében mindig vannak abszolút szélsőértékek, két dimenzió esetén ez nem egyértelmű**

A gondolat egy egyszerűsített problémán könnyen vázolható. Legyen *N* az összes elvégzendő munkaóra, és *M* az alkalmazottak száma, és keressük azt az $n_i$ értéket minden *i*-re 1 és *N* között, ami az *i.* alkalmazott munkaterhe. Könnyen belátható, hogy a legkisebb értékekkel rendelkező pont munkaterhe 1 (hiszen legalább 1 egység munkát el kell végeznie) és *N/M* (hiszen „legrosszabb" esetben mindenki egyenlő mennyiségű munkát végez) közé kell esnie. Ehhez hasonlóan megkapható az is, hogy a legnagyobb munkateherrel rendelkező alkalmazott értékei *N/M* (hasonlóan ha mindenki egyenlő) és $N\text{-}M+1$ (mivel mindenkinek legalább 1 egység munkát el kell végeznie) korlátokkal becsülhető.

Ez az elgondolás egydimenziós terek esetén működőképes volt, és a többi pontra is hasonló jellegű korlátokat, arányosításokat tudtam elvégezni. Sajnos két vagy több dimenzióra nem sikerült absztraktálnom a módszert, és nem életszerű, hogy egyetlen egy paraméter alapján – vagyis esetünkben egyetlen dimenzió alapján hozzunk bármiféle döntést.

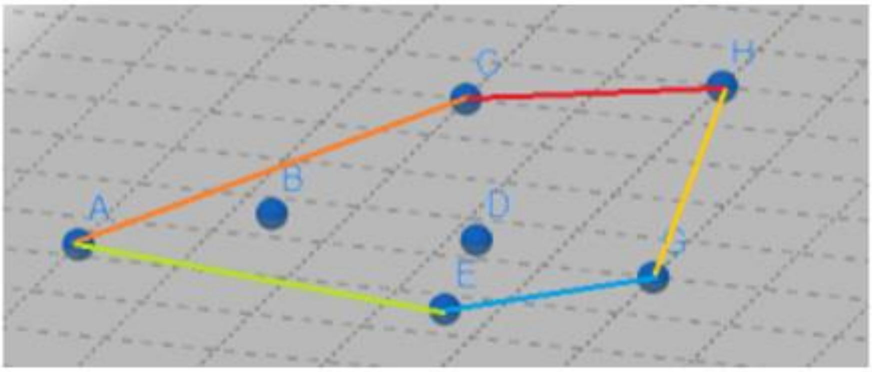
A 7. ábrán szemléltetem ezt a problémát, ahol az egyik dimenzió szélsőértékei A és G, a másiké pedig E és H. A dimenziójukra merőleges egyenesek behúzásával érzékeltetem ezt, ideális esetben lenne két abszolút szélsőérték – ezek az egyenesek által alkotott paralelogramma egy tetszőleges átlójának két csúcsán helyezkednének el.

Ezután megvizsgáltam több arányosító eljárást, szélsőértékkiválasztó módszert, különböző jellegű becslési módszereket, hogy hogyan lehetne ezt a gondolatot több dimenzióra átvezetni, de sajnos egyik módszer sem járt sikerrel,, ezekről a következő alfejezetben értekezem. Mindezek ellenére a gondolat nem volt zsákutca, egy fontos részét később felhasználtam., amire a 3.5 fejezetben ki is fogok térni részletesebben.

### 3.4.2 Konvex burok használata

Logikus továbblépés volt a szélsőértékkeresés többdimenzióra való absztaktálásában a konvex burok. Nagy előnye a konvex buroknak, hogy tetszőlegesen sok dimenzióban egyértelműen előáll. A konvex burok egy geometriai fogalom, ami tetszőlegesen sok dimenzióban történő általánosítással rendelkezik. Egy ponthalmaz konvex burkának azt a minimális méretű konvex alakzatot nevezzük, amely a halmaz összes elemét tartalmazza. Két dimenzió esetében a tartalmazás egyenesekkel, három dimenzóban már síkokkal, felette pedig hipersíkokkal történik.

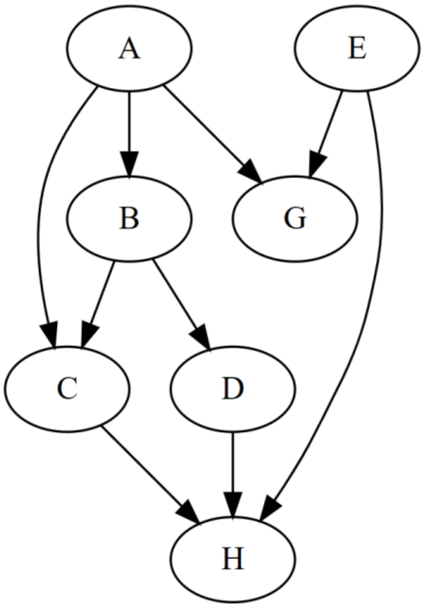
A másik nagy gondolat a konvex burok mögött az volt, hogy ki tudom használni az összes pont egymáshoz viszonyított pozícióját. Megpróbálhatok korlátokat felírni pontokra azáltal, hogy mely pontok zárják közre.




**8. ábra Konvex burok az előző kétdimenziós téren**

A probléma ebben az esetben viszont az volt, hogy nem egyértelmű, hogy a burok egy-egy oldala milyen jellegű korlátként funkcionálhat. Ezután megvizsgáltam a gondolatot, hogy csak valamilyen speciális „alsó illetve felső burkot” számítsak csak ki, ami alsó illetve felső becslésként szolgálhat.

| Jel | Trait I. | Trait II. |
|---|---|---|
| A | 2 | 3 |
| B | 3 | 4 |
| C | 7 | 5 |
| D | 3 | 6 |
| E | 1 | 7 |
| G | 2 | 9 |
| H | 8 | 8 |




**9. ábra Példa bemenet és a hozzá tartozó korlát-gráf**

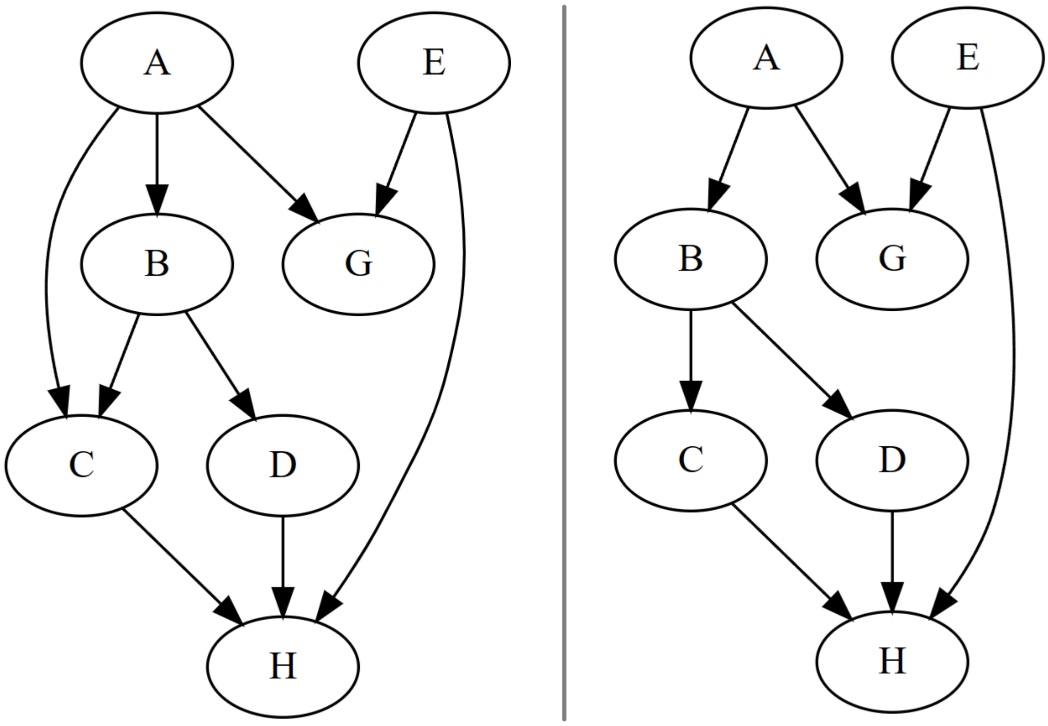
Sajnos ez a módosítás sem vezetett eredményre, viszont a gondolat egy módosított változata később beépítésre került mégis a végleges változatba. A végleges, már gráfokkal működő módszerben ez a gondolat adta a „korlát-gráf” fogalmának ötletét, és erre így is fogok hivatkozni a későbbiekben.

## 3.5 A korlát-gráf használatának alapelvei

A következő gondolat lényege az volt, hogy megvizsgálom, egy adott pont értéke mely többi pontnál biztosan nagyobb, hogy azt majd becslésekhez felhasználhassam. Ezáltal egy irányított $G$ gráfot szeretnék építeni, amit korlát-gráfnak nevezek, ahol $A, B \in v(G)$ csúcsok között akkor és csakis akkor fut irányított él $A$-ból $B$ csúcsba, ha $B$ felülről korlátolja $A$-t. Ez a gyakorlatban azt jelenti, hogy $B$ vektorértékei rendre nagyobbak $A$ megfelelő vektorértékeinél. Könnyűszerrel belátható, hogy így az eredmény egy körmentes gráf a valós számokon értelmezett összehasonlítás tranzitív tulajdonsága miatt.

### 3.5.1 Túl sok felesleges él problémája

Már kevés teszt után is beláttam, hogy az előbb vázolt gráfban nem minden él tartalmaz új információt, viszont a rendszer komplexitását növeli. Legyen *A, B* és $C \in v(G)$, ahol létezik út *A*-ból *B*-be, és *B*-ből *C*-be. Ekkor a korlát-gráf definíciója szerint (szintén az összehasonlítás tranzitivitása miatt) vezetni fog egy él *A*-ból *C*-be is, és ez az él semmilyen új információt nem adott, hisz *B*-vel felülről becsülve egy pontosabb felső korlátot lehet kapni *A*-ra, mint *C*-t használva.




**10. ábra Az előző példa gráfja eredeti állapotában és élelhagyás után**

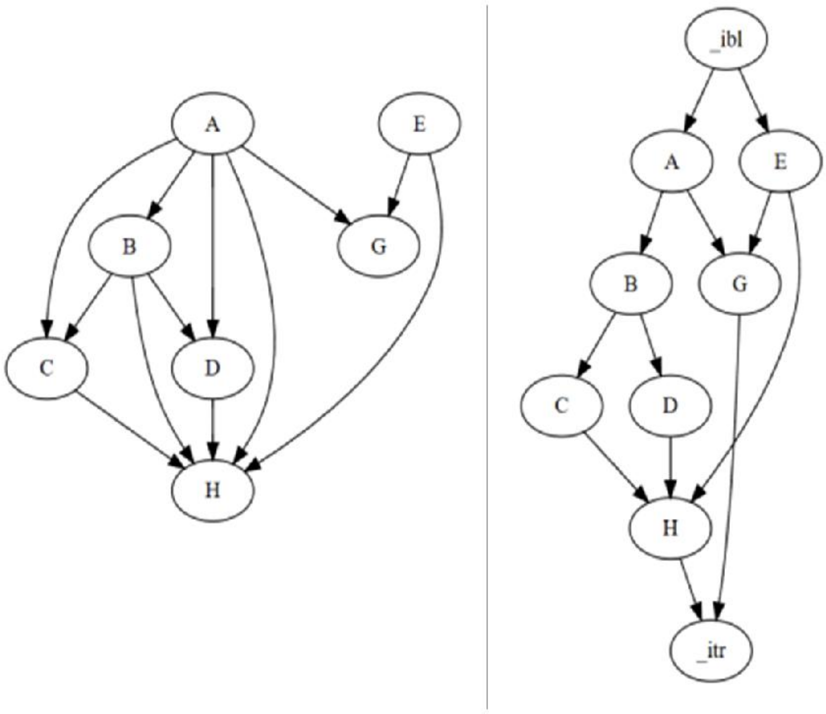
Ezután a gráf kezdeti felépítése után első lépésben azonnal elhagyok minden olyan $A \rightarrow C$ élt, ahol létezik olyan *B* csúcs, hogy legyen $A \rightarrow B$ és $B \rightarrow C$ út is a gráfban. Ahhoz, hogy ezt megtehessem be kell látni először, hogy nem történik információvesztés ezáltal, illetve egy összefüggő gráf megőrzi ezt a tulajdonságát. Ezt az állítást könnyű belátni, hiszen akkor és csakis akkor hagyunk el élt *A* és *C* között, ha egy $A \rightarrow C$ élet fel nem használó, tetszőleges *B* csúcsot érintő úton eljuthatunk *A*-ból *C*-be, ezáltal egy összefüggő gráf mindenképp összefüggő marad.

### 3.5.2 A gráf összefüggőségének problémája

Az előző alfejezetben nagy figyelmet szenteltem arra, hogy a kapott korlát-gráfot a tranzformációk során összefüggőnek tartsam meg, de magyarázatot még nem adtam arra, hogy ez miért volt kiemelten fontos. A következő interpolálandó csúcs kiválasztása a gráfban elfoglalt pozíciója alapján történik, ezért nagyon fontos, hogy ez a gráf összefüggő legyen.

Sajnos ez az összefüggőség nem mindig adódott magától valódi adatokon. Ezen kívül sokszor egynél több bemeneti csúcsunk volt a gráfban, ami szintén problémát okozott. Hasonló helyzet állt fent a felső becslésekkel és a kimeneti csúcsokkal

Ezeknek a problémáknak megoldásaképp felvettem egy-egy imaginárius minimum és maximum csúcsot. A vektoriális értékeik megállapítása egyszerűen történt: minden csúcs minden dimenziójára vettem a minimum illetve maximum értékeket, és ezeket rendeltem hozzá az imaginárius csúcsok megfelelő értékeihez. Ezáltal egy urna alakú (tehát pontosan egy bemeneti és pontosan egy kimeneti csúcsa van), összefüggő, irányított és körmentes gráfot kaptam minden esetben, ahol a bemeneti csúcs az imaginárius minimum, a kimeneti pedig az imaginárius maximum.




**11. ábra Az eredeti gráf az összes tranzformáció után. Az imaginárius csúcsok beillesztését követően pontosan egy abszolút eredő és egy abszolút nyelő csúcsunk lesz**

Ritka esetekben valódi példákban is előállt, hogy volt abszolút minimum vagy maximum érték. Ilyen esetekben nem szükséges felvenni egy új csúcsot, a valódi csúcs használata mint minimum vagy maximum csúcs tökéletes. Olyan ritka esetben, ha megtörténne az, hogy van valódi minimum vagy maximum csúcs, de nem pontosan 1-1 darab, akkor felvehető imaginárius csúcs fiktív értékekkel, mint például a nullvektor, vagy egy vektor mindennél magasabb értékekkel. Ez előbbi miatt dobtam ki a bemeneti adataink közül a nullvektorra alakuló tanárokat, ugyanis ilyen esetben nem képes a modellem imaginárius minimum csúcsot képezni.

Ábrákon és kódban az imaginárius csúcsokat mindig „_" jellel kezdve ábrázolom ezeket a csúcsokat, konvencióként arra az esetre, ha a jövőben több ilyen csúcsom is lenne. Az *„_ibl"* az „imaginary bottom left" rövidítése, és a minimumot jelöli, míg az *„_itr"* az „imaginary top right" kifejezésből származik, utalva a kétdimenziós matematikai koordinátarendszerben elfoglalt pozíciójukra.

Az összes tranzformációs lépés végreahajtását a 11. ábrán szemléltetem. Első lépésként elhagytam a tranzitív tulajdonság miatt fennálló éleket – például az A→C élet – majd pedig egyetlen abszolút eredő és egyetlen abszolút nyelő csúcs hiányában beszúrtam a két imaginárius csúcsot a korábbi abszolút nyelők után, illetve az abszolút eredők elé.

## 3.6 Az interpoláció futása a korlát-gráfon

### 3.6.1 Az első iteráció menete

A következő kérdés az volt, hogy milyen sorrendben kezdjem el interpolálni a gráf csúcsait. Emlékeztetőként fontos megjegyezni, hogy a Lipschitz-interpolációnak két csúcsra van szüksége ahhoz, hogy jó eredményt adjon. Mivel már kiküszöböltem a gráf összefüggőségének problémáját, valamint tudom, hogy minden esetben pontosan egy minimum és maximum csúcsom van, kézenfekvő választás ez a két szélsőérték csúcs. Korábban, az egyenletrendszeres megoldásnál bemutattam, hogy a két szélsőértékcsúcsra tudok egzakt értékkel rendelkező számbecsléseket adni – ezek rendre az 1, *N/M,* illetve az *N-M+1* értékek. A legkisebb hiba elvét használva a minimum értéke az előbbi kettő számtani közepe, míg a maximum utóbbi kettő átlaga lett.

Ezután az interpoláció már rendelkezik két csúccsal, de a következő csúcs kiválasztásának problémája nem lett rendezve. Ekkor a következő gondolat az volt, hogy mindig felezzem meg az intervallumot, amiben a csúcsok elhelyezkednek. Így mindig a legnagyobb intervallum középső csúcsát interpoláltam következőnek, amíg a csúcsok el nem fogytak. Formálisan kifejezve a azt az $A \in v(G)$ csúcsot választottam ki, ahol

$min(succ(A)/(pred(A)+1) \; -1\,)$ fennállt, vagyis ahol a rákövetkező és az őt megelőző még fel nem dolgozott csúcsok aránya a legközelebb áll az 1-hez. A nullával való osztást elkerülendő adtam hozzá 1-et a nevezőhöz, ahol $pred(A)$ értelemszerűen nem lehet -1.

### 3.6.2 A második iteráció menete

Az első iteráció egy nyers becslés a munkateher megállapításához. Ezután egy újabb iterációt végrehajtok, ahol megint minden csúcsra alkalmazom az interpolációt. Itt a korábban említett csúcselhagyásos módszert alkalmazom. Az algoritmus egyszerű, csak veszek minden valódi csúcsot *G* gráfban, és az első iteráció után kapott eredmények alapján – önmagát elhagyva – leinterpolálom.

Fontos megjegyezni, hogy az új értéket nem visszaírom, hanem egy új táblázatba tárolom le, ez már a végleges kimenet lesz. Emiatt könnyen belátható, hogy a kimenete a második iterációnak végrehajtásfüggetlen. Ráadásul az imaginárius szélső csúcsok miatt az sem állhat elő, hogy szélről hagyjunk el csúcsot. Abban a ritka esetben, ha valódi lenne valamelyik szélsőérték csúcs (ami eleve felvet egy puha kérdést, hogy igazságosak-e a paramétereink) több különböző lehetséges módszer alkalmazható. Az egyik ilyen a struccmegoldás, ami jó megoldás lesz, ha a szélsőértékcsúcs be- vagy kifokszáma (szélsőérték jellegéből adódóan) legalább kettő, ugyanis így lesz meredeksége az interpolált függvénynek, és jó értéket kapunk rá. Amennyiben ez sem áll fenn, meg lehet vizsgálni, hogy ahhoz az egy csúcshoz képest milyen *M* meredekséggel kell eltolni a pontunkat ahhoz, hogy a Lipschitz-tulajdonságot megőrizze a ponthalmaz. Ehhez viszont végre kell hajtani a teljes második iterációt, és annak a végeredménye alapján kell ezt a számítást elvégezni.

### 3.6.3 Utófeldolgozás

A második iteráció végrehajtása után viszont nincs garanciám arra, hogy a kapott összmunkateher megegyezik az eredetileg elvárttal. Könnyűszerrel orvosolható ez a probléma. Mindössze annyi dolgom van, hogy „normalizáljam” az értékeket, vagyis az összes kapott munkaterhet megszorozzam az új munkateher szummájának és az eredeti elvárt munkatehernek a hányadosával. Ez a normalizáció miatt viszont valószínűleg tört értékeket fogok kapni, amiknek az alsóegészrészét veszem. Ez a művelet az oka annak, hogy az eljárásom – pár szerencsés kivételtől eltekintve, amikor az interpolált értékek egészek, és a hányados kerek szám – nem paretooptimális, vagyis veszteségmentes.

## 3.7 M kiválasztásának problémája

Többször említettem *M*-et, mint a Lipschitz-tulajdonság egyik legfontosabb paraméterét, valamit többször is felhasználtam számításaim során, de nem magyaráztam még meg azt, hogy M értéke miket vehet fel. Kísérleteim során tapasztalati úton láttam, hogy M 1-nek való választása a legtöbb esetben jó választásnak bizonyult. Minél nagyobb volt 1-nél *M*, annál nagyobb lett az adatok szórása, és minél közelebb került a 0-hoz, annál inkább osztotta szét a munkaterhet az interpoláció szinte egyenlő mértékben.

Az újabb kutatásaimban viszont kipróbáltam több *M* számítást is. Ilyen volt a pontok egydimenziósítása (például a vektorértékeik szummája, produktuma, átlaga), majd ezeknek az értékeknek a szórása. Érdekes eredményeket kaptam így is, de egy másik eljárás jobbnak bizonyult.

Abban a két imaginárius csúcs közötti távolság és a két, az imaginárius csúcsokhoz legközelebb eső 1-1 csúcs távolságának a hányadosát használtam fel, mintegy képzeletbeli átlók arányát *M*-nek. Értelemszerűen minél közelebb esnek a csúcsok az imaginárius csúcsokhoz, annál közelebb lesz a hányadosuk 1-hez. Amennyiben valamelyik szélsőérték valódi, így azt használtam. Érdekes megjegyzés, hogy amennyiben nincs imaginárius csúcsunk, így ez a hányados 1 lesz.

# 4 Eredményeim

Alább mutatom be a gyakorlati eredményeim összevetését egy konkrét implementáció kimenetével. Ezután eredménystatisztikát ismertetek, kitérve a két módszer eredményeinek hasonlóságaira és különbségeire, és ezek okaira.

## 4.1 Összevetés egy fairness through awarness alapján készült eljárással

Kutatásom eredményeit egy beosztástervező algoritmussal vetettem össze, ami a fairness through awarness megközelítést [17] alkalmazta. Fontos megjegyezni, hogy az egy komplett beosztást készített, ezáltal a „puha megszorításokat”, úgynevezett soft constraint-okat is figyelembe vett, és elvégezte az időbeosztást is, ami miatt egy másabb problémát oldott meg. Fontos megjegyezni, hogy gyakorlati lehetetlenség minden soft constraint kielégítése, így az a beosztástervező algoritmus valójában egy minimalizáló problémát old meg, ahol az ideális beosztás a lehető legkevesebb soft constraint-et hágja át, és egy hard constraint-et sem szeg meg.

Viszont a program futásának kimenetéből képes voltam megállapítani az eredmények munkaterhét, amit már össze tudtam vetni a saját eredményeimmel. Az adatokon minimális tranzformációt végre kellett hajtanom, ami annyiban merült ki, hogy a teljesen 0-sorokat eltávolítottam a bemenetből.

Number of input files:1011

| Name | Vizsgáztató hány tárgynál | Össz hallgatószám az oktatott tárgyaknál | Áltagosan hány hallgató jutna rá | Átlagosan hány oktató azoknál a tárgyaknál, amit oktat |
|---|---|---|---|---|
| [redacted] | 2 | 55 | 15 | 4.5 |
| [redacted] | 1 | 20 | 3.3333333333333335 | 6 |
| [redacted] | 2 | 42 | 10.666666666666666 | 4.5 |
| [redacted] | 1 | 20 | 3.3333333333333335 | 6 |
| [redacted] | 1 | 9 | 3 | 3 |
| [redacted] | 2 | 42 | 10.666666666666666 | 4.5 |
| [redacted] | 1 | 22 | 7.333333333333333 | 3 |
| [redacted] | 1 | 35 | 11.666666666666666 | 3 |
| [redacted] | 1 | 35 | 11.666666666666666 | 3 |
| [redacted] | 1 | 1 | 1 | 1 |
| [redacted] | 1 | 20 | 3.3333333333333335 | 6 |
| [redacted] | 1 | 9 | 3 | 3 |
| [redacted] | 1 | 9 | 3 | 3 |
| [redacted] | 2 | 3 | 3 | 1 |
| [redacted] | 1 | 1 | 1 | 1 |
| [redacted] | 1 | 1 | 1 | 1 |
| [redacted] | 2 | 7 | 7 | 1 |
| [redacted] | 1 | 1 | 1 | 1 |

**12. ábra A bementi adatok táblázata. A tanárok neveit személyiségi okokból kitakartam, csak a vezetéknevük és az utolsó utónevük kezdőbetűit hagytam meg.**

Annak az algoritmusnak egyik fontos mérőszáma, célfüggvényértéke, a megszegett soft constraint-ok száma, súlyozva azok fontosságával. 1011 beosztást vizsgáltam meg, amit azzal a módszerrel készítettek. Az 1011 beosztásban a célfüggvényértékek egész nagy intervallumon mozogtak, -100 és -34 között mozogtak, ahol minél közelebb volt egy beosztás célfüggvényértéke a 0-hoz (ami a teoretikus tökéletes beosztás, ámde gyakorlatban nem létezhet a részben ellentmondó soft constraint-ok miatt) annál jobbnak volt nevezve. Az 1011 beosztás ugyanazon az eredeti adathalmaz alapján készült, és különbözőek voltak. Megjegyzésként fontos megjegyeznem, hogy egy meglepően kevés személyből álló beosztás is hatalmas kombinatorikus probléma, így ez az 1011 eset közel sem az összes lehetséges beosztás volt, ami előállhatott az egyszerűsített záróvizsga beosztásra. Viszont mindegyik a maga módján már legalább elfogadhatónak, de inkább jónak nevezhető volt, így valódi minőségű adattal lett össszevetve az eredményem.

## 4.2 Eredmény statisztika

Alább bemutatom az összehasonlítások során érdekesebbnek talált statisztikákat. A próbafutásban vizsgáltam két tanár közötti szélsőséges eltéréseket is, mint ahogyan beosztások közötti (tanárok aggregációjával végzett) különbségeket is. Az eredményekben említett különbségek mértékegysége a munkaóra. Az ábrákon a Python dictionary kulcsai némi magyarázatra szorulhatnak. A target_value kulcs a beosztás célfüggvényértéke, az avg_diff az átlagos különbség a két eljárás által számított tanári órák között. Min- illetve maxdif kulcsokkal jelöltem az adatsor minimális és maximális eltérését, az eltérés „helyét” – tehát a tanárokat, akiknél ez az eltérés előállt – pedig min- illetve maxdifname kulcsok alá szerveztem. A könnyebb összevetés érdekében azokhoz a tanárokhoz letároltam a munkaterhet is az adott adatsorból, ezek az alábbi formátumban voltak: (min|max)dif_(origi|int), ahol az origi az „eredeti”, fairness through awarness eljárást jelöli, az int pedig az interpolációs eljárásra utal. A kényelmesebb visszakeresés érdekében az adatsor filenevét is letároltam.

| | Lowest difference | Highest difference | Lowest average | Highest average | Lowest target value | Highest target value |
|---|---|---|---|---|---|---|
| target value | -84 | -59 | -93 | -88 | -30 | -100 |
| average difference | 4,7 | 5,5 | 3 | 5,7 | 4,5 | 5,1 |
| maximal difference | 15 | 26 | 15 | 16 | 17 | 22 |
| maximal difference position | K_A | K_A | K_A | K_A | K_A | K_A |
| original value for maximal difference position | 25 | 36 | 25 | 26 | 27 | 32 |
| interpolated value for maximal difference position | 10 | 10 | 10 | 10 | 10 | 10 |
| minimal difference | 0 | 1 | 0 | 1 | 0 | 1 |
| Count (number of objects with same examined value) | 494 | 1 | 1 | 1 | 3 | 16 |
| minimal difference position | B_Z | G_B | I_G | G_B | I_G | M_I |
| original value for minimal difference position | 12 | 7 | 8 | 7 | 8 | 3 |
| interpolated value for minimal difference position | 12 | 6 | 8 | 6 | 8 | 4 |

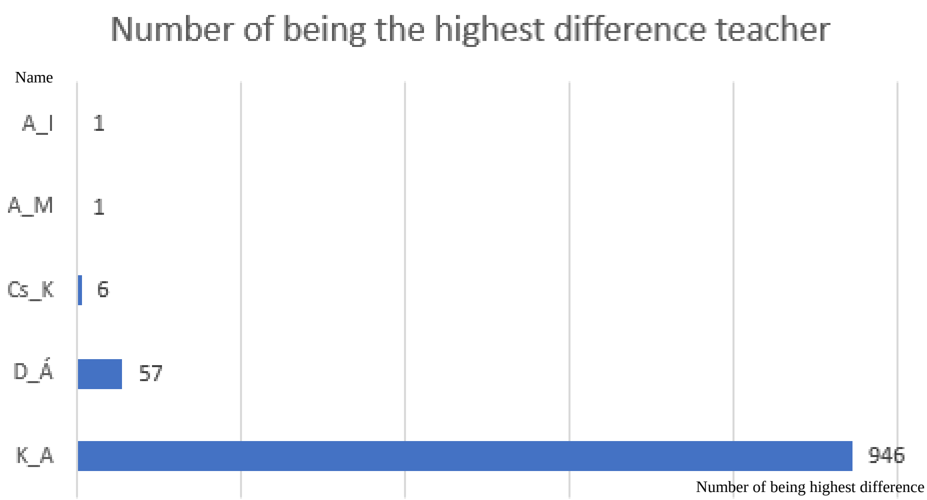
**13. ábra A kimeneti statisztika első része, a nevek a korábban említett módon kitakarva**

Az egyes beosztásokon végzett futtatások során majdnem minden alkalommal volt olyan eset, hogy a minimális munkateher-különbség 0 volt, vagyis volt alkalmazott, akinek az én metrikám alapján megállapított munkaterhe megegyezett azzal, amit a beosztás szerint kapott. Átlagosan a minimális munkateher eltérése a két eljárás között ~0,5 volt, ez is mutatja, hogy legalább egy alkalmazottra a munkateher gyakran – egészen konkrétan 494 alkalommal – megegyezett.

Az legmagasabb eltérés már jóval nagyobb volt, ami a két megközelítés merőben más paradigmáira hívja fel a figyelmet. Egyetlen alkalommal az interpolációalapú és a fairness through awarness megközelítés különbsége 26 volt. A célfüggvényérték arra a beosztásra -59 volt, ami a célfüggvényértékek eloszlása alapján a generált beosztások között egy gyengébbnek mondható volt – de továbbra is persze egy életszerű, jó beosztás volt.

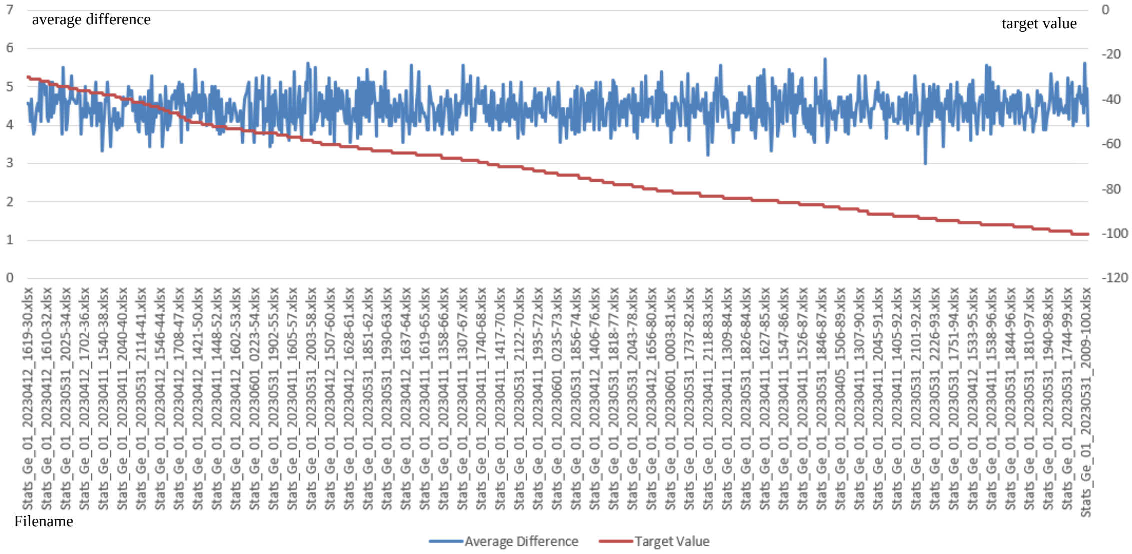
A legmagasabb átlagos eltérés viszonylag alacsony volt, mindössze ~5,7. Ez a beosztás azon beosztások egyike, ahol a minimális eltérés sem 0 volt – szám szerint 1 -, a maximális eltérés pedig 16, ami, habár nagy, jelentősen kisebb a korábban említett 26-nál. Ennek a beosztásnak a célfüggvényértéke -88 volt, ami kifejezetten gyenge eredménynek számít a többi beosztás célfüggvényéhez képest.

Ami az őszinte meglepetésemre szolgált, az az, hogy a legkisebb átlagos eltéréssel bíró adatsor célfüggvényértéke is kifejezetten magas volt, egészen konkrétan ez az érték arra a beosztásra -93 volt. Itt az átlagos eltérés mindössze 3 volt, ami szerintem kifejezetten alacsony. Ami még ennek a bemenetnek az érdekessége, hogy a maximális különbség itt is egészen magas volt, 16.




**14. ábra Összesített statisztika a legnagyobb különbség személyéről**

A fairness through awarness eljárás által készített legjobb beosztás – melynek célfüggvényértéke -30 volt – átlagos különbsége nem volt sokkal magasabb, mint a legalacsonyabb átlagos különbség, mindössze ~4,5 volt.




**15. ábra Az átlagos különbség (elsődleges tengely) és célfüggvényérték (másodlagos tengely) nem korrelált**

Készítettem összesített statisztikát is az 1011 bemeneti adatról. Átlagoltam a különbségeket, illetve megvizsgáltam a legtöbbször előálló legmagasabb különbséget okozó tanárokat. K_A kiugróan sokszor szerepelt, így elkezdtem vizsgálni, hogy ennek mi lehetett az oka. Arra jutottam, hogy ezt a különbséget a gyenge megkötések okozzák. Ugyanis K_A számos konzultált hallgatóval rendelkezett, amit az én modellem nem vett figyelembe (mint ahogy a konzultációt magát sem, hiszen a konzulensi jelenlét nem kötelező). Viszont a másik modellben egy preferencia volt, hogy a konzulens valamilyen beosztásban szerepeljen a záróvizsga során. Mivel K_A sok konzultált hallgatóval rendelkezett, és ütközésmentesen beosztható volt, így a többi számadata alapján elvártnál nagyobb mértékben került beosztásra. D_Á többszörös feltűnésére számítottam korábban, ugyanis amikor az egy ember terheltségét vizsgáltam ütközőhármasok segítségével, akkor (másféle) bemenet esetén többször előfordult a tesztfuttatásaim során. Ott megvizsgálva az értékeket, arra jutottam, hogy D_Á egyszerűen több vizsgát vállal, mint paraméterei alapján logikus volna.

# 5 Kutatásom összegzése, téma továbbvitele

Bemutattam kutatásom eredményét, az interpolációalapú igazságvizsgálatot, és annak egy konkrét használati területét beosztástervezéseken. A modellem kellően általános és rugalmas ahhoz, hogy különböző problémákra, problémakörökre átültethető legyen a jövőben, ezt pedig mindenképp érdemes lehet megvizsgálni. Összevetettem az eljárásom elméleti alapjait más, az irodalomban népszerű igazságvizsgálati irányzatokkal, amik alapján egy újfajta igazságdefiníciót alkottam. Ezután pedig eredményeim összehasonlítottam egy másik eljárással, amik alapján állítható, hogy eljárásom felhasználható beosztások igazságának vizsgálatára, képes igazságtalan elbánásokat megtalálni, ráadásul akár egy konkrét beosztás elkészítéséhez is mankót adhat.

Az eljárásom hatalmas előnye, hogy az öt igazságvizsgálati axiómából olyan négyet használ fel, amire korábban nem volt elterjedt, ismert példa. Ez a fajta megközelítés miatt akár más igazságvizsgálati metrikákkal együtt is lehet alkalmazni, hogy egy átfogóbb, pontosabb eredményt kapjunk egy beosztásról. Hatalmas előnye még az eljárásomnak a transzparencia, a számítások teljes mértékben történő átláthatósága.

Úgy gondolom, hogy ebben a gondolatban nagy potenciál rejlik. Az eredmények tovább javíthatóak azáltal is már, ha több dimenziót választok ki, illetve a kiválasztott dimenziókon finomítok. Mindenképpen érdemes lehet megvizsgálni az interpolációban felhasznált M konstans másféle megállapításának módjait is, ami közvetlenül azzal jár, hogy az eredményeim javulni fognak, pontosabb beosztásokat fog készíteni az eljárás.

# 6 Irodalomjegyzék


[1] J. Zou és L. Schiebinger, „AI can be sexist and racist — it's time to make it fair | Nature," 2018. [Online]. Available: https://www.nature.com/articles/d41586-018-05707-8.

[2] J. E. Fountain, „The moon, the ghetto and artificial intelligence: Reducing systemic racism in computational algorithms," *Government Information Quarterly,* 2022.

[3] C. Dwork, M. Hardt, T. Pitassi, O. Reingold és R. Zemel, „Fairness Through Awareness," *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference,* 2012.

[4] K. Eriksson, D. Estep és C. Johnson, „Lipschitz Continuity," in *Applied Mathematics: Body and Soul,* Springer, Springer, 2004.

[5] S. Verma és J. Rubin, „Fairness definitions explained," *Proceedings of the International Workshop on Software Fairness,* 2018.

[6] G. Beliakov, „Interpolation of Lipschitz functions," *Journal of Computational and Applied Mathematics,* 2006.

[7] G. Lulli és A.Odoni, „The European Air Traffic Flow Management Problem," *Transportation Science,* 2007.

[8] V. Prasnikar és A. E. Roth, „Considerations of Fairness and Strategy: Experimental Data from Sequential Games," *The Quarterly Journal of Economics,* 1992.

[9] A. E. Roth, T. Sönmez és M. U. Ünver, „Kidney Exchange," *The Quarterly Journal of Economics,* 2004.

[10] H. SHI, R. V. Prasad, E. Onur és I. Niemegeers, „Fairness in Wireless Networks:Issues, Measures and Challenges," *IEEE Communications Surveys & Tutorials,* 2013.

[11] J. F. Nash, „The Bargaining Problem," *ECONOMETRICA,* 1950.

[12] S. Martello, „Jenő Egerváry: from the origins of the Hungarian algorithm to satellite communication,” *Central European Journal of Operations Research,* 2009.

[13] R. J. Vanderbei, Linear Programming, 2020.

[14] D. Nace és M. Pioro, „Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial,” *IEEE Communications Surveys & Tutorials,* 2008.

[15] D. Bertsimas, V. F. Farias és N. Trichakis, „The Price of Fairness,” *Operations Research,* 2011.

[16] H. Kameda, E. Altman, C. Touati és A. Legrand, „Nash equilibrium based fairness,” *Math Meth Oper Res,* 2012.

[17] E. Kalai és M. Smorodinsky, „Other solutions to Nash's bargaining problem,” *Econometrica,* 1975.

[18] X. L. Huang és B. Bensaou, „On max-min fairness and scheduling in wireless ad-hoc networks: analytical framework and implementation,” *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing,* 2001.

[19] E. Hahne, „Round-robin scheduling for max-min fairness in data networks,” *IEEE Journal on Selected Areas in Communications,* 1991.

[20] A. Sridharan és B. Krishnamachari, „Maximizing network utilization with max–min fairness in wireless sensor networks,” *Springer, Wireless Networks,* 2009.

[21] F. P. Kelly, A. K. Maulloo és D. K. H. Tan, „Rate control for communication networks: shadow prices, proportional fairness and stability,” *Journal of the Operational Research Society,* 1997.

[22] L. B. Jiang és S. C. Liew, „Proportional fairness in wireless LANs and ad hoc networks,” *IEEE Conference on Wireless Communications and Networking,* 2005.

[23] L. Li, M. Pal és Y. R. Yang, „Proportional Fairness in Multi-Rate Wireless LANs,” *IEEE Annual Joint Conference: INFOCOM, IEEE Computer and Communications Societies,* 2008.

[24] S. Erdős és B. Kővári, „Genetic Algorithm Based Solution for Final Exam Scheduling," *MultiScience - XXXIII. microCAD International Multidisciplinary Scientific Conference,* 2019.