



EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

INFORMÁCIÓS RENDSZEREK

TANSZÉK

## Általánosított Euler-diagramok automatikus elrendezése

*Témavezető:*

Dr. Molnár Bálint

habilitált egyetemi docens

*Szerző:*

Sarkadi-Nagy Bence

programtervező informatikus MSc

*Budapest, 2020*

Az eredeti szakdolgozati / diplomamunka témabejelentő helye.

# Tartalomjegyzék

<b>1. Absztrakt</b>	<b>3</b>
<b>2. Elméleti alapok</b>	<b>4</b>
2.1. A dolgozat felépítése . . . . .	4
2.2. Probléma . . . . .	5
2.2.1. Alapdefiníciók . . . . .	5
2.2.2. Gráfok ábrázolása . . . . .	8
2.2.3. Hipergráfok és halmazrendszerek ábrázolása . . . . .	9
2.2.4. Euler-diagramok vizualizációja . . . . .	12
2.2.5. Problémaleírás . . . . .	16
2.3. Megoldás . . . . .	17
2.3.1. Optimalizáció . . . . .	17
2.3.2. Függvényapproximáció . . . . .	22
2.3.3. Mesterséges neurális hálók . . . . .	22
2.3.4. Gráf konvolúciós neurális hálózatok . . . . .	25
2.3.5. Hipergráf konvolúciós neurális hálózatok . . . . .	26
<b>3. Saját eredmények</b>	<b>28</b>
3.1. Vizsgált módszerek . . . . .	28
3.1.1. Euler-diagram reprezentáció . . . . .	28
3.1.2. Optimalizációs módszerek . . . . .	31
3.1.3. Inicializációs módszerek . . . . .	35
3.1.4. Heurisztikák . . . . .	37
3.1.5. Összefoglalás . . . . .	45
3.2. Mérések és következtetések . . . . .	46
3.2.1. Mérési környezet . . . . .	47

3.2.2. Érzékenység a probléma paramétereire . . . . .	49
3.2.3. Algoritmusok hatékony paraméterezésére . . . . .	54
3.2.4. Előnyös inicializálási módszerek . . . . .	55
3.2.5. Heurisztikák . . . . .	59
3.2.6. Hipergráf konvolúciós hálózat . . . . .	62
3.2.7. Futásidők . . . . .	62
3.2.8. Összefoglalás . . . . .	62
<b>4. Konklúzió</b>	<b>63</b>
<b>Irodalomjegyzék</b>	<b>64</b>

# 1. fejezet

## Absztrakt

A hipergráfok gyakran alkalmazott matematikai eszközök az informatika számos területén, így a szemantikus web, bioinformatika, szenzorhálózatok, adatbázisrendszerek, szociális hálózatok, gépi látás és egyéb részterületek számos megoldása épül rájuk. Ezen feladatok vizsgálata során különösen hasznos lehet a hipergráfok ábrázolása. Vizualizáció során egyszerre merül fel igény a mögöttes struktúra tökéletes leképezésére, a könnyű értelmezhetőségre (így esztétikai és ergonómiai metrikákra), az általános alkalmazhatóságra, gyors futásidőre és - ezzel összefüggésben - a megjeleníthető adathalmaz méretének maximalizálására is, továbbá gyakran felmerül a dinamikus környezet feladatak kezelésének problémája is.

Mint látható, ez egy felettesebb összetett problémát eredményez, amely megoldására számos módszer született már a szakterületi irodalomban, azonban ezek gyakran szenvednek egy – vagy több – tervezési elv sérülésétől, így egyik sem terjedt el a gyakorlatban. Jelen dolgozatban különböző optimalizációs módszerek, illetve heurisztikák teljesítményét vizsgáljuk a fenti szempontok alapján, különös tekintettel az általános alkalmazhatóságra és a szemantikailag helyes leképezésre.

## 2. fejezet

### Elméleti alapok

#### 2.1. A dolgozat felépítése

Segítendő a szövegben való tájékozódást, ebben az alfejezetben részletezem a dolgozat felépítését. A szöveg két fő fejezetre bontható, amelyek előtt egy rövid összefoglaló (Absztrakt), után pedig egy összefoglaló fejezet (Konklúzió) található. Az első (Elméleti alapok című) fejezetben először megadom a probléma leírásához, illetve megértéséhez szükséges definíciókat és fogalmakat, majd a tágabb és szűkebb problématerület irodalmi áttekintésére kerül sor, az általam kitűzött pontos feladat megfogalmazásával lezárva. Az első fejezet második felében a megoldás során alkalmazott módszerek elméleti alapjait tekintem át. A második fejezet (Saját eredmények) az általam vizsgált különböző megoldási módszereket írja, illetve ezek teljesítményét vizsgálja különböző paraméterek mellett, továbbá a levont következtetésekre is itt térek ki.

**Megjegyzés.** *A dolgozatban szereplő egyes szakszavak (néha egész szakterületek) egyáltalán nem, vagy nem elég hangsúlyosan szerepelnek a magyar szakirodalomban ahhoz, hogy elterjedt fordításuk legyen. Ezekben az esetekben – további figyelmeztetés nélkül – az angol nyelvű megfelelőjüket fogom használni. Népszerű, szinonímaként használt szakkifejezéseket igyekszem per jellel elválasztva felsorolni.*

## 2.2. Probléma

### 2.2.1. Alapdefiníciók

Ahhoz, hogy megértsük a megoldandó problémát, számos fogalmat át kell tekintsünk először.

#### Gráfok

A gráf (variációi) alapvető fontosságú adattípus(ok) a modern informatikai megoldások során. Definiálásuk nem csak a dolgozat későbbi részeiben felbukkanó algoritmusok miatt fontos, hanem a – gyakran a gráfok általánosításának tekintett – hipergráfok mélyebb megértését is elősegíti.

**1. Definíció.** *Irányítatlan gráfnak* nevezünk egy olyan, rendezett  $G = (V, E)$  párt, ahol  $V$  egy nem-üres halmaz,  $E$  pedig egy olyan multihalmaz, amely a  $V$  elemeiből képzett kétélemű halmazokat tartalmaz. Formálisan  $E \subseteq \{\{u, v\} | u, v \in V\}$ . A  $V$  halmazt *csúcshalmaznak* is szokás nevezni, elemeit *csúcsoknak*, míg  $E$ -t az *élhalmaz*, elemeit pedig *él* névvel illetjük. Az élek által tartalmazott elemeket az adott él *végpontjainak* hívjuk. Két csúcs *szomszédos*, ha van olyan él  $G$ -ben, amely őket tartalmazza, míg egy csúcs *izolált*, ha egyetlen elnök sem végpontja. A  $G' = (V', E')$ ,  $V' \subseteq V$ ,  $E' \subseteq E$  a  $G = (V, E)$  gráf *részgráfja*.

**2. Definíció.** Egy nem-üres halmazból,  $V$ -ból, és az elemeiből képzett rendezett párokat tartalmazó  $A$  multihalmazból képzett rendezett  $D = (V, A)$  párt *irányított gráfnak* hívjuk. Az irányítatlan gráfok nömenklatúrája itt is érvényes, azonban az élek rendezett párájában az első elemet speciálisan *kiindulópontnak*, a másodikat pedig *végpontnak* is szokás nevezni. Azt mondjuk, hogy egy  $(u, v) \in A$  él az  $u$  csúcsnak egy *kimenő éllel*,  $v$ -nek egy *bemenő éllel*. *Forrásnak* nevezzük azt a csúcsot, amelynek nincsenek bemenő élei, *nyelőnek* azt, aminek nincsenek kimenő élei.

**3. Definíció.** Az  $e_i, e_j \in E$ ,  $i \neq j$  éleket *párhuzamos éleknek* nevezzük, ha  $e_i = e_j$ . *Hurokélnék* egy olyan élet nevezünk, amely  $\{v, v\}$  vagy  $(v, v)$  formájú, azaz a két végpontja azonos.

**4. Definíció.** A  $v \in V$  él **fokszámát**  $d$ -vel jelöljük, azaz  $d = |\{e|e \in E \wedge v \in e\}|$ . A  $G$  gráf  **$k$ -reguláris**, ha minden csúcsának fokszáma  $k$ . Irányított gráf esetében megkülönbeztetjük a **befokszámot** és a **kifokszámot**.

**5. Definíció.** **Egyszerű gráf** egy olyan irányítatlan gráf, amelyben sem párhuzamos, sem hurokélek nincsenek jelen.

**6. Definíció.** **Páros gráfnak** hívunk egy  $G = (V, E)$  gráfot, ha  $V = A \cup B$ ,  $A \cap B = \emptyset$  és sem  $A$ -n, sem  $B$ -n belül nem fut él. Jelölése:  $G = (A, B)$  vagy  $G = (A, B, E)$ .

**7. Definíció.** Irányított gráf esetén élek egy  $(u_1, v_1), \dots, (u_k, v_k)$  sorozatát **sétának** nevezzük, ha  $v_i = u_{i+1}$ ,  $i = 1 \dots k - 1$ . Irányítatlan gráfok esetén analóg módon definiáljuk a fogalmat, azonban eltekintünk a csúcsok élen belüli sorrendjétől. Ha a séta semelyik két éle nem tartalmazza ugyanazt a csúcsot, akkor a séta **útnak** mondjuk. Ha egyedül a séta kezdőpontja egyezik meg a végpontjával, akkor az egy **kör**. Két csúcs **távolsága** a köztük lévő legrövidebb útban szereplő élek száma, ha ilyen nincs, akkor végtelen. Egy  $v$  csúcs **elérhető** az  $s$  csúcsból, ha a távolsága nem végtelen tőle. **Összefüggőnek** mondott egy gráf (vagy részgráf), ha abban minden csúcs elérhető mindegyik másikból, míg egy **összefüggő komponens** a vizsgált gráf csúcsainak egy olyan részhalmaza, amely összefüggő, de nem bővíthető úgy, hogy az maradjon.

**Megjegyzés.** Irányítatlan gráfokban az elérhetőség ekvivalens az összefüggőséggel, így lineáris időben megkapható. (A linearis futásidő elérhető irányított gráfokra is.)

**8. Definíció.**  $K_n$ -nel jelöljük az  $n$  csúcsú egyszerű gráfot, amelyben minden csúcspár között fut él, az ilyen gráfok neve **teljes gráf**. Mikor csak egy részgráfra igaz ez a tulajdonság, akkor azt **teljes részgráfnak** vagy **klikknek** mondjuk.

**9. Definíció.** **Topologikus sorrendként** ismert a  $D = (V, A)$  irányított gráf csúcsainak egy olyan sorrendje, ahol minden csúcsból csak nála nagyobb sorszámuva megy él. A topologikus sorrend megléte ekvivalens azzal, hogy az adott gráfban nincsen kör, így az ilyet **körmentes gráfnak** nevezzük.

**10. Definíció.** Egy  $G = (V, E)$  irányítatlan gráf **line graphja** alatt az  $L(G) = (E, \{\{e_i, e_j\} | \exists v \in V : v \in e_i \wedge v \in e_j, i \neq j\})$  irányítatlan gráfot értjük.

## Halmazrendszerek és hipergráfok

Mint az absztraktban is említettem, halmazalapú adatreprézentációkkal, így hipergráfokkal és halmazrendszerekkel az informatika számos területén találkozhatunk. Az egyes problématerületek – sőt, gyakran az egy problématerületet vizsgáló különböző szakcikkek – azonban egymással konkuráló definíciókat alkalmaznak a hipergráfokra. Sokszor a halmazrendszerek szinonímájának tekintik a fogalmat, míg például – a szakterület egyik alapművének számító – Hypergraphs c. [1] kötet mind az általunk használt – mindenki megismertetett – definícióhoz, mind a halmazrendszer alapúhoz képest megszorításokat vezet be.

**11. Definíció.** A  $H$  halmaz **hatványhalmaza**  $\mathcal{P}(H) = \{x | x \subseteq H\}$ , azaz a  $H$  halmaz összes részhalmazainak halmaza.

**12. Definíció.** A  $H$  halmaz fölötti  $\mathcal{F}$  halmazrendszert  $\mathcal{F} \subseteq \mathcal{P}(H)$ -ként definiáljuk.

**13. Definíció.** Ebben a dolgozatban **irányítatlan hipergráf** vagy egyszerűen **hipergráf** alatt egy olyan, rendezett  $H = (V, E)$  párt értünk, ahol  $V$  a **(hiper)csúcsok** nem-üres halmaza, míg  $E$ , a **hiperélek** halmaza, egy olyan multihalmaz, amelynek az elemei  $\mathcal{P}(V) \setminus \emptyset$ -ből kerülnek ki.

**Megjegyzés.** A hipergráf, úgy is ismertek, mint a gráfok általánosításai, ahol minden hiperél pontosan 2 elemet tartalmaz, azaz 2-reguláris. Egyrészről ez jól láthatóan függ a választott gráf-, és hipergráfdefiníciótól. Az itt használt definíciók alapján hipergráfok nem tartalmazhatnak hurokéléket, viszont párhuzamos éléket igen, így nem tökéletes általánosításai az irányítatlan gráfoknak, viszont irányítatlan egyszerű gráfoknak már igen.

**Megjegyzés.** Egy másik felmerülő kérdés a hipergráfok kapcsán a hiperutak, más szóval a tranzitivitás fogalma a hiperélek között. A szakirodalomban erre is több, azonban jobban elkülönülő definíció létezik. Az itt bemutatott eredmények nem építenek a tranzitív relációra, így tetszőleges definíció tételezhető fel.

## Gráf- és hipergráf-adatszerkezetek

Gráfok gépi kezelésére köztes gráfreprézentációkra, gráfadatszerkezetekre van szükség. Leggyakrabban az adjancia mátrix, az incidencia mátrix, az éllista és a

ritka reprezentációk általános osztálya használatos. Mivel a megoldások során csak az első kettőt alkalmazzuk, ezért a többöt itt nem is definiálom.

**14. Definíció.** A  $G = (V, E)$  gráf **adjacencia mátrixa** alatt azt a  $|V| \times |V|$  méretű  $A_G$  mátrixot értjük, amelyben  $a_{ij} = 1$ , ha  $\{v_i, v_j\} \in E$ , egyébként 0.

**15. Definíció.** A  $G = (V, E)$  gráf **incidencia mátrixa** alatt azt a  $|V| \times |E|$  méretű  $I_G$  mátrixot értjük, amelyben  $i_{jk} = 1$ , ha  $v_j \in e_k$ , egyébként 0.

**16. Definíció.** A  $H = (V, E)$  hipergráf **incidencia mátrixát** ugyanúgy definiáljuk, ahogy a gráfokon értelmezett megfelelőjét.

**Megjegyzés.** Hipergráfok esetében az adjacencia mátrix nem értelmezett.

**17. Definíció.** A  $H = (V, E)$  hipergráf **line/intersection graphja** az  $L(H) = (E, \{\{e_i, e_j\} | e_i \cap e_j \neq \emptyset, i \neq j\})$  képlettel kapott egyszerű gráfot fedi. Jól látható, hogy ez általánosítása a gráfok esetében alkalmazott definíciónak.

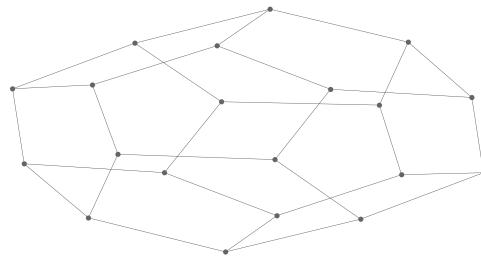
**18. Definíció.** A  $H = (V, E_H)$  hipergráf **bipartite incidence structure-e** alatt azt a  $G = (A, B, E_G)$  párós gráfot értjük, amelyben  $A = V, B = E_H$  és  $E_G = \{\{u, v\} | u \in V, v \in E_H, u \in v\}$ .

**19. Definíció.** Egy irányítatlan  $H = (V_H, E_H)$  hipergráf (**szuper**)duálisa az a  $G = (V_G, E_G)$  egyszerű gráf, amelyben  $V_G = \{X | X \subseteq E_H \wedge \exists v \in V_H : ((\forall e \in X : v \in e) \wedge (\forall e \notin X : v \notin e))\}$  és  $E_G = \{\{X, Y\} | X, Y \subseteq E_H, X \neq Y \wedge \exists v \in V_H : (v \in X \wedge v \in Y)\}$

**Megjegyzés.** Figyeljük meg, hogy a szuperduálisban szereplő csúcsok száma exponentiális az eredeti csúcshalmaz tekintetében!

### 2.2.2. Gráfok ábrázolása

Mivel a gráfok gyakran használt, könnyen konceptualizálható modellek, ezért a számítógépes vizsgálat mellett gyakran előnyös a felhasználó számára vizualizálni őket. Gráfok reprezentálhatók halmazokként vagy numerikusan (például adjacencia mátrixként), azonban ember-gép interakciók során a leggyakrabban olyan képként szokás ábrázolni őket, amelyen a csúcsok köröknek, az őket összekötő élek pedig egyenes (egyes esetekben akár görbe) vonalaknak felelnek meg.



2.1. ábra. Egyszerű gráf képi ábrázolása

Gyakorlatban különösen fontos tulajdonságnak bizonyult – a könnyű értelmezhetőség szempontjából –, hogy egy adott gráf ábrázolható-e anélkül, hogy bármely két éle metszené egymást, azaz *síkbarajzolható-e*. A gráfelmélet egy ismert tétele, hogy nem minden gráf rajzolható síkba. A **Fáry-Wagner** tétel továbbá kimondja, hogy minden olyan gráf, ami görbe vonalak használatával síkbarajzolható, az egyenes vonalak esetén is az marad.

Annak ellenére, hogy a reprezentáció aránylag egységes, rengeteg módszer létezik a konkrét csúcselrendezési feladat megoldására (ami természetesen implikál egy teljes gráfkirajzolást is). Ezen módszerek aztán csoportosíthatók, így ismerünk egyszerű konstruktív szabályokon (például köríven vagy rácson való elhelyezésen) alapulókat[2, 3, 4], a Laplace-mátrix sajátvektorait használó spektrális elrendezéseket[5] és – jellemzően – rúgóként kezelt élek és elektronokként tekintett csúcsok egyensúlyi állapotát kereső erőalapú (force-directed) módszereket[6, 7]. A mi szempontunkból ezutóbbi lesz a legfontosabb, azon belül is Fruchterman és Reingold megoldása[8]. Természetesen a korábbiak nem minden különülnek el teljesen, előfordul, hogy több kategóriába is sorolható egyazon algoritmus[9].

### 2.2.3. Hipergráfok és halmazrendszerek ábrázolása

Gráfokhoz hasonlóan hipergráfok (és/vagy halmazrendszerek) esetén is hasznos a felhasználó által értelmezhető ábrázolásuk, azonban – a gráfoktól eltérő módon – a vizuális reprezentációjuk már egyáltalán nem egységes. A szakirodalom számos különböző módszert tart nyilván, melyeket Alsallakh és társai foglaltak össze[10]. Ugyan Alsallakhék számos kategóriába sorolták a módszereket, azonban – véleményem szerint – kis variációk is könnyen összemossák az ezek közti határokat. Bár

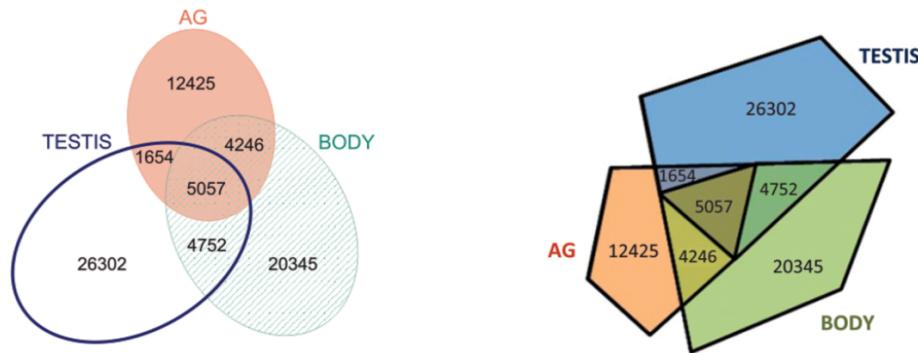
én általánosabb kategóriák szerint tekintem át a módszereket, ezek sem tekinthetők abszolútnak, gyakori a hibridek használata.

A legtöbb cikk régióalapú módszereket alkalmaz, ahol az egyes halmazok és egymáshoz való viszonyuk a sík részterületeivel vannak reprezentálva, míg a régiók elkülönítése jellemzően színkódolással történik[11, 12, 13] (az ebben a fejezetben bevezetett Euler-diagramok is ebbe a kategóriába tartoznak). Egy másik népszerű eszköz a gráf szerű leírás. Ide tartozik például a halmazok görbevonalként való leírása, amelyek akkor tartalmaznak egy csúcsot, ha átmennek azon[14]. Elképzelhető továbbá a szuperduális használata is, amely végső soron tökéletesen reprezentál egy hipergráfot, azonban a tartalmazás reláció nehezebben felismerhető válik tőle. A bipartite incidence structure használata esetén ez a probléma már nem merül fel, viszont Skiena kivételével – aki minden össze meglemlíti az alkalmazhatóságát[15] – nem találtam olyan cikket, amely közvetlenül használná. Természetesen mátrixokkal (pl. incidencia mátrix segítségével) is ábrázolható egy hipergráf. Különböző aggregációt használó algoritmusok is előfordulnak, amelyek leginkább méretarányos diagramok vagy felhasználói interakció bevezetésével operálnak[16, 17], habár ezek nem különösen elterjedtek. Ennek ellenére Chapmanék empirikus értékelése szerint az általuk használt aggregációs módszer könnyebben és jobban értelmezhető volt, mint a népszerűbb, Euler-diagramokon alapuló megoldások[16]. Előfordul továbbá, hogy információt ikonokkal írnak le, speciálisan a régióalapú megoldásoknál glyphnek nevezük, mikor az egyes metszetek fölötti ikon(ok) méretével vagy számával jelezzük azok elemeinek számát.

Ebben a dolgozatban egy specifikus, régióalapú ábrázolási mód, az Euler-diagram vizsgálatát tűztem ki célul. Az Euler-diagram kirajzolását célzó algoritmusok az úgynevezett Euler Diagram Generation Problem (EDGP) megoldásai. Ahogy a neve is mutatja, a szóban forgó ábrázolási módot még maga Leonhard Euler vezette be a XVIII. században, azonban mindmáig előszeretettel használatos. Ahogy Baron írja[18], Euler minden össze példákon mutatta be, illetve alkalmazta módszerét, nem definálta explicit módon. Ezek alapján – a hipergráfokhoz mintájára – Euler-diagramokra is többféle definíció létezik, melyek minden meggyeznek abban, hogy az egyes halmazok/hiperélék zárt görbekkel reprezentáltak, melyek metszetei közös halmazelemeket feltételeznek, míg diszjunkt esetben azok hiányát. Egyes definíciók az

alaphalmaz elemeit/hipercsúcsokat is elhelyezik az ábrán, így létrejöhetnek olyan metszetek is a vizualizáció során, melyek valójából üresek, de az értelmezés során – elviekinben – ez mégsem okoz gondot. Más esetekben üres részhalmazok létrejötte nem engedélyezett vagy egy speciális színnel jelölendő. Gyakori kérdés, hogy a zárt görbek körök, ellipszisek vagy tetszőleges formájúak lehetnek-e, hogy szükségszerűen konvexek-e, illetve az egyes halmazok több, azonos címkével ellátott görbével is reprezentálhatók-e. A következőkben definiálom, hogy ebben a dolgozatban milyen értelmezést használok, azonban – az előzőeknek megfelelően – egyéb források ettől eltérhetnek.

**20. Definíció.** A továbbiakban a  $H = (V, E)$  hipergráfot reprezentáló **egyszerű Euler-diagram** alatt egy olyan ábrát értünk, amelyben minden  $e \in E$  halmaz egyetlen zárt görbére képződik le, mely azokat, és csak azokat a hipercsúcsokat tartalmazza, melyek az adott  $e$  hiperélnek is elemei. Azon hipercsúcsok, melyek egyetlen hiperélben sem szerepelnek, az összes görbén kívül kell megjelenjenek. Az egyes görbek címkével (esetünkben színekkel) rendelkeznek.

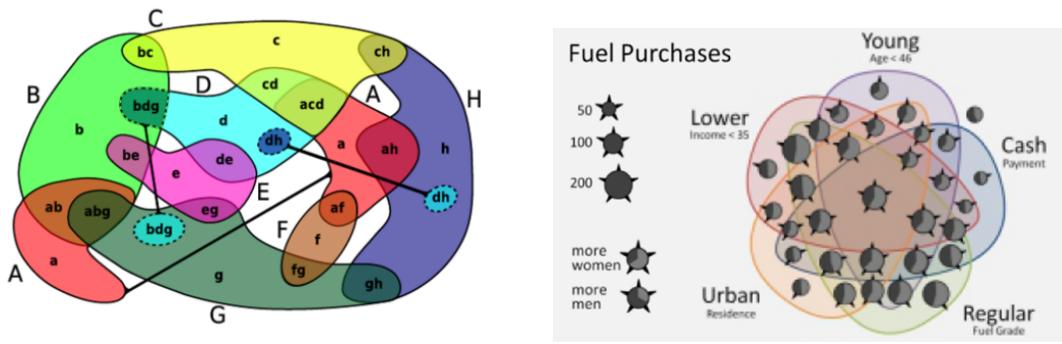


(a) Ellipszisalapú Euler-diagram[11]

(b) Poligonalapú Euler-diagram[12]

2.2. ábra. Egyszerű Euler-diagramok

**21. Definíció.** Általánosított **Euler-diagramként** fogom nevezni azt az egyszerű Euler-diagrammot, mely egy hiperelt több, azonos címkével ellátott zárt görbére is leképezzet.



(a) Általánosított Euler-diagram[13]

(b) Hibrid Euler-diagram glyphekkel[19]

2.3. ábra. Euler-diagram variánsok

#### 2.2.4. Euler-diagramok vizualizációja

##### Síkbarajzolhatóság

Gráfok esetében láthattuk, hogy felmerül a síkbarajzolhatóság kérdése. Egy hipergráf akkor síkbarajzolható, ha a kirajzolása során nem jön létre olyan régió, amely üres halmazt reprezentál. Amennyiben egyszer Euler-diagramokkal, így zárt, a végpontokat (jelen esetben hipercsúcsokat) tartalmazó görbékkel reprezentáljuk az éleket, akkor rögtön láthatjuk, hogy ezeknek tartalmazniuk kell legalább egy görbét is, amely közvetlenül összeköti a végpontokat. Ebből már észrevehetjük, hogy csúcsok használata nélkül nem minden hipergráf rajzolható síkba egyszerű Euler-diagramokkal.

Verroust és Viaud bebizonyította[20], hogy az általuk használt Euler-diagram definíció 8 halmazig megőrzi a síkbarajzolhatósági tulajdonságot. Simonetto és Auber egy másik struktúrát vizsgált[13], ami megfelel az itt általánosított Euler-diagramként definiált fogalomnak (ők Euler representationnek nevezik), melyről levezették, hogy alkalmas tetszőleges hipergráf síkbarajzolására, görbéknek kizárálag szemantikailag helyes metszeteinek létrehozása mellett (akár csúcsok használata nélkül is). A szuperduális felhasználásával pontos leírása is adható annak, hogy mikor nem rajzolható ki egy hipergráf egyszerű Euler-diagramok használatával[13, 20]. (Megjegyzendő, hogy a Simonetto cikk intersection graphnak nevezi a szuperduális, miközben az egy másik struktúrát jelöl, de a leírásból, illetve a példákból levezethető, hogy valójából felcserélték a kettőt.)

## Esztétikai mértékek

Természetesen egy diagram nem csak a mögöttes struktúra (halmazrendszer vagy hipergráf) leképezéséhez feltétlenül szükséges szemantikai tulajdonságokkal bírhat. Egy adott ábrának az emberi értelmezést segítő tulajdonságait esztétikai vagy ergonómiai tulajdonságoknak nevezzük, ha pedig számszerűsíthetők (és adott rajtuk egy teljes rendezés), akkor esztétikai metrikáknak hívjuk. Esztétikai metrikák definiálhatók magukon a görbéken, a görbek metszetein, a csúcsok eloszlásán, az ábra színezésén, továbbá – gyakorlatilag – az ábrán megjelenő bármely egyéb aspektus fölött [21, 22, 23].

Az EDGP során felmerülő két leggyakrabban vizsgált[16, 21, 22, 24, 25] esztétikai tulajdonság, az úgynevezett well-formed és well-matched tulajdonságok.

**22. Definíció.** *Egy adott Euler-diagram esetén kontúr/contour névvel illetjük az egy címkéhez (vagy hiperélhez/halmazhoz) tartozó különböző zárt görbek összességét. Minimális régiónak/minimal regionnek hívjuk a görbek és metszeteik által létrehozott legkisebb síkpartíciót, míg alaprégió/basic region alatt azon minimális régiók halmazát értjük, melyek ugyanazon görbek részei. Egy zóna/zone az alaprégiók egy olyan halmaza, amelyeket azonos címkével rendelkező görbek tartalmaznak.*

A well-formed tulajdonság hat kritériumból tevődik össze, bár néha csak az első ötöt használják:

1. minden görbe egyszerű, azaz nem metszi önmagát. - WFC1
2. Nincs két görbe, amelynek közös határolószakasza van. (Az elfajuló, pontbeli találkozást nem vesszük hozzá.) - WFC2
3. Nincs olyan pont, ahol három görbe érintkezik. - WFC3
4. Ha két görbe érintkezik, akkor metszik egymást. - WFC4
5. minden zóna összefüggő, azaz egyetlen minimális régióból áll. - WFC5
6. Nem rendelkezik két görbe azonos címkével. - WFC6

Egy Euler-diagram well-matched, ha a zónák, görbek, minimális régiók és kontúrok szintjén is az. A különböző esetekben ezek az alábbiakat fedik:

1. Zónák szintjén: nem tartalmaz üres zónákat. - WMP1

2. Görbék szintjén: a halmazok közti részhalmaz, metszet és diszjunkció relációk megfelelnek az adott halmazokat reprezentáló görbék tartalmazás, átfedés és diszjunkció tulajdonságának. - WMP2
3. Minimális régiók szintjén: well-matched a zónák szintjén, és csak összefüggő zónákat tartalmaz. - WMP3
4. Kontúrok szintjén: a halmazok közti részhalmaz, metszet és diszjunkció relációk megfelelnek az adott halmazokat reprezentáló kontúrok tartalmazás, átfedés és diszjunkció tulajdonságának. - WMP4

Szintúgy széles körben vizsgált tulajdonság az area-proportionality, avagy méretarányosság[26, 27, 28, 29], amely azt mondja ki, hogy minden régió (bizonyos definíciók szerint az univerzumot reprezentáló kivételével) mérete úgy aránylik az ilyenek összegéhez, mint az  $\omega$  súlyfüggvényük azok összegéhez. Leggyakrabban a régió által tartalmazott elemek számát alkalmazzuk súlyfüggvényként. Hibamértek segítségével könnyen metrika is előállítható a tulajdonságból.

Egyes szerzők a régiók színezését is hasonlóan fontosnak ismerik el, mint az elhelyezkedésüket[30, 31].

Annak ellenére, hogy több esztétikai metrika és tulajdonság is széles körben alkalmazott, nagyon kevés empirikus tapasztalatunk van arról, hogy ezek ténylegesen befolyásolják-e egy ábra értelmezhetőségét. Fish és társai kis mintán vizsgálták[32] a well-formed tulajdonságnak az ábrák megértésére vonatkozó hatását. Az ő eredményeik alapján a WFC2 megsértése akár segítheti is egy ábra értelmezését, míg a WFC1 és WFC4 egyidejű, illetve a WFC3 vagy WFC4 önálló megsértése is rontja azt. Blake és társai[25] arra az eredményre jutottak, hogy az egyes görbék orientációja nincs hatással az emberi percepciójukra. Blake-ék, egy másik cikkükben[33], a szimmetrikus alakzatokat azonosították a legkönyebben megérthetőként, így különösen a kör alakú reprezentációt javasolják. Chapman és társai empirikus módon arra az eredményre jutottak[16], hogy a well-matched tulajdonság fontosabb, mint a well-formed.

### Euler-diagramok generálása

Még úgy is, hogy az Euler-diagramok minden részterületét képezik a halmazábrázolási módszereknek, a szakirodalomban rengeteg különböző eljárás található,

melyek gyakran nem is tekinthetők ugyanannak a szűken vett probléma megoldásának. Az alkalmazott Euler-diagram definíció, a vizsgált probléma mérete, az alkalmazott esztétikai tulajdonságok és metrikák, illetve ezek erős vagy gyenge megkövetelése mind-mind új variánsait hozzák létre a – összefoglaló néven EDGP-nek nevezett – problémának.

A – már említett – Allsalakh cikk[10] nem kizárálag a szakirodalomban előforduló EDGP módszereket mutatja be, de ezeket különböző szempontok szerint össze is hasonlítja. Elsősorban megkülönbözteti a tetszőleges relációk ábrázolására alkalmas, illetve az ebben a tekintetben korlátozott megoldásokat. Ettől nem függetlenül megadják, hogy milyen alakzatokkal reprezentál egy halmazt az adott módszer (kör, poligon vagy ellipszis), a cikkből azonban sajnos kimaradt, hogy ismert Bézier-görbe alapú megoldás is[23]. Másik szempontként hozzák fel a teljesített esztétikai tulajdonságokat, mint a well-formed, well-matched, area-proportional, szimmetrikus görbe tulajdonságokat, és vizsgálják, hogy létrejönnek-e üres minimális régiók (az univerzumon kívül). Az általuk adott összehasonlítás alapján továbbá azt tételezhetjük fel, hogy az egyes módszerek vagy három, vagy tetszőleges számú halmazra alkalmazhatók. Megfigyelhető, hogy ezutóbbi az alapján válik el, hogy az Euler-diagramok egy speciális esetét, a Venn-diagramokat vizsgálja-e egy adott cikk vagy az általános problémát. Amiről ezek alapján nem kapunk képet, hogy egyes módszerek csak 8 halmazig alkalmazhatók[20], mivel ezek fölött már ismertek olyan példák[13, 20, 34], amelyek nem síkbarajzolhatók egyes Euler-diagram definíciók szerint. (A cikkben nem említett, de hasznos kiemelni, hogy egyes algoritmusok csak már meglévő diagramok esztétikai javítását szolgálják[21] vagy emberi beavatkozást igényelnek[35].)

A dolgozat későbbi részeiben legfontosabbnak Flower, Rodgers és Mutton munkája[23] fog bizonyulni, akik sztochasztikus optimalizációs módszereket, illetve metaheurisztikákat alkalmaztak Bézier-görbékkel reprezentált Euler-diagramokra, és akikkel részben hasonló megközelítést választottunk. Érdemes még megemlíteni Stapleton és társainak munkáját[34], amelyben induktív módon generálnak well-formed euler diagramokat, mikor ez lehetséges, a többi esetben pedig a well-formed kritériumok megsértésével, a görbék önmetszésével érik el, hogy továbbra is szemantikailag helyes ábrát generáljanak. Különösen érdemes megfigyelni, hogy az általá-

nosság ilyen szintű eléréséhez a duális gráf egy módosított verzióját használják, ami exponenciális futásidőt eredményez.

### 2.2.5. Problémaleírás

Láthattuk, hogy az EDGP egy összetett probléma, amely magában foglalja a konkrét ábrázolási mód (Euler-diagram definíció), a vizsgált esztétikai metrikák, az elfogadható futásidő és a megoldható problémaméret meghatározását is. Különösen fontos kitérni arra is, hogy egy adott probléma vizsgálata során nem feltétlenül egyfélre szempont szeretnénk ábrázolási módot választani, elképzelhető például, hogy ugyanúgy szeretnénk az előforduló klasztereket vizsgálni, mint a leghosszabb utakat, melyek másféle elrendezést tételeznek fel.

Ebben a dolgozatban az abszolút általános EDGP megoldás felé igyekszem egy lépést tenni, amennyiben – direkt módszerekkel szemben – többféle esztétikai metrika kezelésére is képes optimalizációs módszerek alkalmazhatóságát vizsgálom általánosított Euler-diagramok esetén. A szakirodalomnak megfelelően egy-két tucat hiperelág és legfeljebb száz csúcsig vizsgálom a problémát. Mivel ez egy kísérletsorozat, ezért a futásidőre megkötést nem teszek, azonban az eljárások kidolgozása során probléma méretében exponenciális futásidő megoldásokat kizárom.

Különösképp megnehezíti a feladatot az általános megoldás igénye, illetve Alsallakhék (a halmazábrázolási módszereket áttekintő cikkükben[10] tett) megállapítása, miszerint az Euler-diagramok mindössze 10-20 halmazig alkalmazhatók.

Mint láthattuk, az általános megoldhatóság érdekében emberi beavatkozás, korlát nélküli paramétertér (például Bézier görbék kontrollpontjai[23]) vagy exponenciális futásidő[34] lehet szükséges. Mivel tetszőleges esztétikai metrika fölött az optimalizáció, így a legjobb Euler-diagram megtalálása is NP-nehéz, ezért ez egyáltalán nem meglepő. A megoldásom alapjaként választott Flower cikkel[23] szemben, az általam használt, bonyolultabb problémater (mind a hipergráf méretében, egy adott halmazt reprezentáló zárt görbék számában és ebből kifolyólag a költségfüggvényként alkalmazott heurisztikák nem-folytonos jellegében) felveti a diagram modell egyszerűsítésének igényét, illetve újabb heurisztikák kidolgozásának szükségességét.

Vizsgálataimat ezért megelőzi egy diagram modell, a rászabott – de a szakirodalmi megállapításokon alapuló – heurisztikák és az optimalizációs algoritmusok egyedi

variánsainak kialakítása.

## 2.3. Megoldás

### 2.3.1. Optimalizáció

A matematikai optimalizáció célja egy adott  $f : X \rightarrow \mathbb{R}$  valós értékű függvény globális minimum- vagy maximumhelyének megtalálása, ahol  $X$  tetszőleges halmaz lehet. Mivel az  $f$  függvény negáltja segítségével maximalizációs problémák minimalizáliós problémákká vezethetők vissza (vagy fordítva), ezért a kettő feladatot azonosnak tekintjük. A továbbiakban – mikor nem jelzem az ellenkezőjét – minimalizálási problémát tételezek fel.

**23. Definíció.** Az  $f$  függvényt számos módon nevezik; minimalizálási problémák esetén **költség-/veszteségfüggvényként** vagy **objektívfüggvényként**, míg maximálizálási problémák esetében **utility** vagy **fitness functionként** ismerjük. Egyes szakterületeken (mint fizikában) egyéb nevek is ismertek. Ha az  $f$  függvény a  $g_i, i \in \mathbb{N}_0$  függvények átlagaként áll elő, akkor költségfüggvénynek hívjuk, a  $g_i$  függvényeket pedig veszteségfüggvényeknek.

**24. Definíció.** Egy optimalizálási algoritmus paramétereit **hiperparamétereknek** nevezzük.

Általánosságban beszélve az optimalizálási probléma NP-nehéz, azonban a költségfüggvényre tett megszorításokkal ez feloldható. Különösen fontos ezért, hogy az egyes algoritmusok, hogy milyen megszorítások mellett operálnak.

### Gradiensalapú optimalizáció

Gradiens-, illetve deriváltalapú algoritmusok vagy a deriváltfüggvényt (Hessemátrixot) vagy a pontbeli (parciális) deriváltat alkalmazzák. Bár szigorúan véve az első- és másodikderivált-próba is ide tartozik, a gyakorlatban a vizsgált függvény pontos képlete jellemzően nem ismert, illetve a paraméterek és a lokális szélsőértékek nagy száma is ellehetetleníti ezt a féle megoldást. Ezzel szemben iteratív módszereket szokás használni, melyek egy megadott – általában véletlenszerű – kiindulópontból egy lokális minimumponthoz konvergálnak. Amennyiben a függvény konvex vagy

a kiindulópont kellőképpen közel volt a globális minimumhoz, akkor a kapott lokális minimumhely globálisan is az lesz, azonban ez általában nem garantált. Az előzőeknek megfelelően sokszor elvárt tulajdonság a folytonos deriválhatóság is.

A gyakorlatban használt legtöbb iteratív algoritmus a gradiens leszálláson alapszik. Gradiens leszállás során egy tetszőleges  $\theta_0$  kiindulópontot választunk, majd a  $\theta_{i+1} = \theta_i - \alpha * \nabla f(\theta_i)$  szabály alapján kiválasztjuk a következő vizsgálandó pontot, ahol  $\alpha$  egy hiperparaméter. Az iteráció addig folytatódik, míg a megállási feltétel (például felső korlát az iterációk számára, a lépésköz nagyságára vagy a gradiens egy normájára) nem teljesül. Fontos tulajdonsága ennek az algoritmusnak, hogy túl nagynak megválasztva  $\alpha$ -t átugorhatunk minimumhelyeket, míg túl kicsire állításával a keresési időt növeljük meg. Számos módszer született ennek a hiányosságnak az áthidalására, így egyes variációk az iterációk számának növekedésének függvényében csökkentik  $\alpha$ -t vagy eltérő kiindulópontokból indítják újra a keresést. Bizonyos feltételek mellett az algoritmus garantáltan egy lokális minimumhelyhez konvergál[36].

A gradiensalapú módszerekkel rokon jegyeket mutat a – nem feltétlenül differenciálható, de konvex függvények esetében a alkalmazható – szubgradiensmódszer, illetve az általánosabban használható, lokális keresésen alapuló hegymászó algoritmus.

## Sztochasztikus algoritmusok

Összefoglaló néven **sztochasztikus algoritmus** névvel illetjük azokat a módszereket, amelyek futásuk során valószínűségi változókat alkalmaznak. Ez a megközelítés gyakran használt, mikor a költségfüggvény nem alkalmas direkt optimalizációra, viszont megelégszünk közelítő megoldásokkal is. Szintúgy előnyös, mikor a keresési térben számos, globális minimumértékhez közel álló pont létezik. A gradiens leszállás során látott véletlen kezdőpont kiválasztását inputnak tekintjük, így az algoritmus alapverzióját nem tekintjük sztochasztikusnak, viszont – mint számos egyéb determinisztikus (nem sztochasztikus) algoritmusnak – vannak sztochasztikus variánsai. Ennek megfelelően fontos kiemelni, hogy az egyes optimalizációs módszereknek általában több variánsa is létezhet, melyek esetenként összemossák a kategóriákat. Ennek a fejezetnek a további részében át fogjuk tekinteni azokat az algoritmusokat, amelyeknek szükséges az ismerete a továbbiakban, azonban néhol pont az itt

bemutatott alapverziótól való eltérés lesz különösen érdekes.

A legismertebb ilyen variáns az úgynevezett **sztochasztikus gradiens leszálás** (SGD). Mikor az  $f$  költségfüggvény több  $f_i$  veszteségfüggvény átlagaként jön létre, azaz  $f(x) = 1/n * \sum_{i=1}^n f_i(x)$  (például gépi tanulás során több mintaelemen alkalmazva ugyanazt a veszteségfüggvényt), akkor előfordul, hogy a gradiens leszálás során a mai számítógépek kapacitásához képest túl sok parciális deriváltat kéne kiértékelni. Az SGD ezt úgy kerüli el, hogy a veszteségfüggvényeket egyesével értékeli ki, mindegyik kiértékelés után modósítva a paramétereket. Láthatóan ez nem ekvivalens a teljes költségfüggvény gradiensének használatával (csak közelíti azt), továbbá nagyban függ a kiértékelés sorrendjétől is, ezért a veszteségfüggvények sorrendje a kiértékelés során randomizált. Néha szintúgy SGD-nek nevezik a mini-batch gradient descent módszert, ami annyiban tér el az SGD-től, hogy a paraméterek módosítása során egyszerre  $k$  darab veszteségfüggvény átlagát használjuk, ahol  $k$  egy hiperparaméter.

Természetben lejátszódó folyamatok számos esetben optimalizációs módszerként is tekinthetők. A természeti megfigyelések ből adaptált optimalizációs technikákat összefoglaló néven **biológiaiag inspirált** algoritmusoknak nevezzük. A természeti rendszerek komplexitásából fakadó modellezési bizonytalanság feloldását sokszor a sztochaszticitás bevezetésével érjük el, így – gyakorlatban – a legtöbb biológiaiag inspirált optimalizációs algoritmus egyben sztochasztikus algoritmus is.

Egy ilyen, természet ihlette algoritmus, a természetes szelekcióon alapuló **genetikus algoritmus** (GA)[37, 38]. A módszer mögötti megfontolás, hogy egy populáció életciklusa a új egyedek születéséből/halálából, az egyedek szaporodásából (így genetikai keveredéséből), illetve mutációkból tevődik össze, ahol a párosodás a gének előnyös jellegének  $f$ -nek függvénye. Az algoritmus célja, hogy ezt az  $f$  fitness függvényt maximalizálja, az életképes egyedek kombinációjával, illetve kis perturbációkkal.

---

### 1. Algoritmus Genetikus algoritmus (saját szerkesztés)

**Function** GA(*populationSize*, *selectionRate*, *mutationRate*)

- 1:  $t = 0$
  - 2:  $population_t = \text{generateFeasibleSolutions}(populationSize)$
  - 3:  $values = \text{evaluateFitness}(population_t)$
  - 4:  $bestFitness, bestPosition = \text{updateBest}(values, \infty, population_t, bestPosition)$
-

---

```
5: while megállási feltétel nem teljesült do
6:   parents = selectFitnessProportionally(populationt, values, selectionRate)
7:   children = recombine(parents)
8:   populationt = deleteLast(populationt, values, |children|)
9:   populationt+1 = populationt ∪ children
10:  populationt+1 = mutate(populationt+1, mutationRate)
11:  values = evaluateFitness(populationt+1)
12:  bestFitness = updateBest(values, bestFitness, populationt+1, bestPosition)
13:  t = t + 1
14: end while
15: return bestFitness, bestPosition
```

---

Itt a  $generateFeasibleSolutions(n)$   $n$  darab megengedett megoldást generál,  $evaluateFitness(population)$  kiértékeli az egyes egyedek fitness értékeit,  $updateBest(newValues, oldBest, newPositions, oldPosition)$  az új értékek és pozíciók, illetve a korábbi legjobbak segítségével kiválasztja a legjobbat (maximumot és maximumhelyet),  $selectFitnessProportionally(population, fitnessValues, num)$  a fitness értékek által implikált valószínűségi eloszlás szerint kiválaszt  $num$  darab egyedet,  $recombine(population)$  két elemenként egy új egyedet hoz létre, amely a szülei génnálományán alapszik,  $deleteLast(population, fitnessValues, num)$  törli a  $num$  darab legrosszabb fitness értékkel bíró egyedet a populációból,  $mutate(population, mutationRate)$  pedig a  $mutationRate$  arányában megváltoztatja az egyedek génnálományát. A megállási feltétel gyakran iterációs felső korlát vagy alsó korlát a legjobb fitness értékre.

A **particle swarm optimization** (PSO) egy másik biológiaiag inspirált algoritmus, amely analóg módon működik egyes rajként együtt dolgozó állat- és rovarfajokkal[38, 39, 40]. A módszer alapelve, hogy különböző, részecskeknek nevezett, entitások pozícióiban értékeljük ki az  $f$  költségfüggvényt. A részecskek haladási irányával és sebességgel rendelkeznek, amelyet minden iterációban úgy frissítünk, hogy - véletlenszerű mértékben - figyelembe vesszük magát az irányt/sebességet és a részecske által, illetve a globálisan talált eddigi legjobb pozíciót/értéket.

---

**2. Algoritmus** Particle swarm optimization (Clerc alapján[41])

---

**Function** InitializePSO( $S, lowerBounds, upperBounds, f$ )

```

1:  $bestPosition_{global} = \emptyset$ 
2:  $bestValue = \infty$ 
3:  $bestValues[S] = \text{empty array}$ 
4: for  $i = 1 \dots S$  do
5:    $bestPosition_i = particlePosition_i \sim U(lowerBounds, upperBounds)$ 
6:    $bestValues_i = f(bestPosition_i)$ 
7:    $velocityRange = |upperBounds - lowerBounds|$ 
8:    $velocity_i \sim U(-velocityRange, velocityRange)$ 
9:   if  $bestValues_i < bestValue$  then
10:     $bestPosition_{global} = bestPosition_i$ 
11:     $bestValue = bestValues_i$ 
12:   end if
13: end for
14: while megállási feltétel nem teljesült do
15:   for  $i = 1 \dots S$  do
16:      $random_1, random_2 \sim U([0, 1]^{\dim(lowerBounds)})$ 
17:      $velocity_i = w * velocity_i + c_1 * random_1 * (bestPosition_i - particlePosition_i) +$ 
         $c_2 * random_2 * (bestPosition_{global} - particlePosition_i)$ 
18:      $particlePosition_i = particlePosition_i + velocity_i \quad \text{néha } \dots + \alpha * velocity_i$ 
19:      $particleValue = f(particlePosition_i)$ 
20:     if  $particleValue < bestValues_i$  then
21:        $bestPosition_i = particlePosition_i$ 
22:        $bestValues_i = particleValue$ 
23:       if  $bestValues_i < bestValue$  then
24:          $bestPosition_{global} = bestPosition_i$ 
25:          $bestValue = bestValues_i$ 
26:       end if
27:     end if
28:   end for
29: end while
30: return  $bestValue, bestPosition_{global}$ 

```

---

Ahogy látható algoritmus hiperparméterek segítségével adja meg a részecskék számát, illetve a módosítási szabály egyes tagjainak súlyát.

### Egyéb optimalizációs módszerek

Természetesen az áttekintett kategóriákat nem merítettük ki, számos további algoritmus és variáns áttekintésére a dolgozat keretei között nincsen lehetőségem, illetve további megszorítások és függvényosztályok is ismertek, amelyekre létezik optimalizációs algoritmus, mint például lineáris egyenlőtlenségrendszer vagy diszkrét halmazok esetén.

#### 2.3.2. Függvényapproximáció

A numerikus analízis – így például az optimalizáció – számos területén merül fel a – jellemzően valós értékű – matematikai függvények használatának az igénye. Mikor a vizsgált függvény túl komplex, nem mindenhol kiértékelhető, esetleg nem rendelkezik az elvárt tulajdonságokkal, akkor függvények egy meghatározott részhalmazából kiválaszthatunk egy az eredetire legjobban illeszkedőt. Ezt az eljárást – mint az ezt vizsgáló szakterületet – **függvényapproximációnak** nevezzük. Mikor meghatározott pontokban várjuk csak el a legjobb illeszkedést, akkor görbék illesztéséről beszélünk.

Legyen  $f = (f_1 \dots f_n)^\top, w \in \mathcal{R}^n$ , ahol  $f_i, i = 1 \dots n$  tetszőleges függvény lehet. Ekkor az  $f(x, w)$  **lineáris függvényapproximátornak** nevezzük, ha az lineáris súlyok  $w$  vektorában (bár nem feltétlenül az az  $x$  inputban), azaz  $f(x, w) = w_1 * f_1(x) + \dots + w_n * f_n(x)$ . Mikor az  $f$  függvény nem teljesíti a linearitási feltételt, akkor **nemlineáris függvényapproximátorról** beszélünk.

#### 2.3.3. Mesterséges neurális hálók

Az egyik legismertebb és legszélesebb körben alkalmazott nemlineáris függvényapproximátorok a **mesterséges neurális hálók** (artificial neural network - ANN). Hasonlóan a biológiaiag inspirált algoritmusokhoz, ezek olyan számítási rendszerek, amelyek biológiai folyamatokat, speciálisan az emberi - illetve állati - agyban működő neuronokat, és azok működését modellezik. Egy ANN legkönnyebben irá-

nyított gráfként képzelhető el, amelyben (**mesterséges**) **neuronok** alkotják a valós számokkal címkézett csúcsokat, míg a köztük létező, súllyal rendelkező kapcsolatok (szinapszisok) az éleket. A csúcsok címkéjét **biasnek** hívjuk, az élekét **weightnek**. A weight és a bias értékek közösen adják ki a neurális hálózat paramétereit. Az így kapott gráf számítási rendszerként fogható fel, amennyiben az egyes csúcsok műveleteket reprezentálnak. A forráscsúcsoknak közvetlenül beadhatók a rendszer bemenetei, míg az összes többi csúcs egy újabb értéket számít ki, melyek a nyelő csúcsokban értelmezhetők végeredményként. A források kivételével az egyes csúcsok az  $x_v = \sigma(w_n \cdot x_n + b_v)$  képlet alapján számítják ki az értéküket, ahol  $x_v$  a  $v$  csúcs új értéke (nem címkéje),  $w_n$  a bemenő élek súlyainak (weight) vektora,  $x_n$  ezen élek kiindulópontjainak értéke,  $b_v$  a  $v$  csúcs címkéje/biase, a  $\sigma$  függvény pedig egy - jellemzően nemlineáris - aktiválási függvény.

## Architektúra

A számítási gráf csúcsait általában a forrásuktól való távolságuk alapján szokás ún. **rétegekbe** partionálni, ahol strukturálisan az egyes rétegek általában minden csúcsukban azonos módon viselkednek. A rétegekre bontás számos előnyös tulajdon-sághoz vezet. Egyrészt a különböző rétegek szemantikailag egyre magasabb szintű absztrakciókat reprezentálnak, másrészt rétegen belül párhuzamosan is kiszámít-hatók. Mivel a rétegek minden csúcsukban hasonlóan épülnek fel, ezért gyakran megkülönböztetünk speciális célú rétegeket. Ilyen az időkomponenssel bíró adatok kezelésére kifejlesztett rekurrens/visszacsatolt réteg, a lokális információkat összeg-ző konvolúciós réteg vagy az általánosabb, teljesen összekötött réteg. A rétegekből (vagy anélkül) kialakított számítási gráfot a neurális hálózat architektúrájának is szokás nevezni.

Amennyiben a rétegek szekvenciálisan egymásra épülnek (azaz általában), akkor az egyes rétegek weight és bias értékeit könnyebben kezelhető mátrix alakban is felírhatjuk. Jelöljük az  $i$ -edik réteget  $H^{(i)}$ -vel, a rákövetkezőt  $H^{(i+1)}$ -gyel, az adott rétegen található csúcsok számát pedig  $|H^{(j)}|$ -vel. Ekkor az  $i$ -edik rétegből az  $i+1$ -edikbe menő élek súlyai egy  $W^{(i)} \in \mathcal{R}^{|H^{(i)}| \times |H^{(i+1)}|}$  mátrixban tárolhatók, amelyben – értelemszerűen – a sorok jelentik az élek kiindulópontját, az oszlopok pedig a végpontját. Egy  $B^{(i)} \in \mathcal{R}^{|H^{(i)}|}$  bias vektor is kialakítható, amely a csúcsok

címkéjét tartalmazza. Ezek felhasználásával egy réteg kimenetének kiszámítása a  $H^{(i+1)} = \sigma(H^{(i)} * W^{(i)} + B^{(i)})$  egyenletre redukálódik, ahol a  $\sigma$  függvény az elemenként alkalmazott aktiválási függvényt jelöli, a többi művelet pedig mátrixműveleteket. Ebben a formában a bemenetet a nulladik réteg kimenetének tekintjük. Egyes speciális rétegek ettől a számítási módtól néha kis mértékben eltérnek (például konvolúciót alkalmaznak mátrix szorzás helyett).

## Tanulás

Neurális hálózatok architektúrája leggyakrabban emberi munkával készül el. Van példa arra is, hogy jól működő (a célnak megfelelő pontosságú eredményt adó) neurális hálózatok paramétereit is emberi erővel vagy egyszerű konstruktív szabályokkal állítanák be, azonban ezek a megoldások szélsőséges esetekben, kis méretű hálózatokon szoktak működni. A paraméterek automatikus konfigurálását szokás a hálózat tanításának nevezni, míg az erre alkalmazott algoritmus paramétereit hiperparaméternek. Számos ilyen algoritmus létezik, azonban manapság a sztochasztikus gradiens leszállás variánsait szokás használni. Az SGD alapú tanulás során véletlenszerű módon inicializáljuk a paramétereket, ami után a tényleges tanulás a kimenetre alkalmazott költségfüggvény segítségével történik, melynek függvényében módosítjuk a hibás (rossz eredményre jutó) paramétereket. Különösen nehéz értékelni, hogy egy rossz kimenethez egy adott paraméter mennyire járult hozzá, azonban itt nem térünk ki az erre alkalmazott módszerekre. A tanításnak lehetnek másodlagos követelményei is, mint például a kapott paraméterek komplexitásának (értékskálájának) csökkentése. Fontos kiemelni, hogy nem mentesek a gradiensalapú megoldások a problémáktól (kezdve a paraméter/kimenet hozzárendelési problémától, a szükségszerűen differenciálható költségfüggvényen át, a deriválhatósági szempontból megfelelő inicializálási módszerekig bezárólag), és egyáltalán nem csak ilyen tanulóalgoritmusok léteznek. Evolúciós algoritmusok hatásfokát a neurális hálózatok tanításának kontextusában már számos szerző vizsgálta, illetve javította[42, 43]. Bár ígéretes eredmények érhetők el genetikus algoritmusokkal, és nem is szükséges hozzá, hogy deriválható költségfüggvényt alkalmazzunk, de – egyelőre – ez a módszer nem terjedt el széles körben, azonban – neuroevolúció néven – virágzó szakterületté nőtte ki magát a problémakör.

### 2.3.4. Gráf konvolúciós neurális hálózatok

Említés szintjén láttuk, hogy lokális információk összegzésére konvolúciós rétegek alkalmasak. Mivel gráfokon létezik távolságfogalom, ezért elméletben a körükben is alkalmazható ugyanez az elv. A sztenderd konvolúciós hálózat bemenete azonban egy olyan tenzor (azaz, a mi szempontunkból, egy tetszőleges dimenziószámú mátrix), amelyben a lokalitás a tenzor egyes elemei közti indextávolságon alapszik. Az általunk látott gráfprezentációk (incidencia mátrix, adjacencia mátrix stb.) izomorf gráfot írnak le akkor is, ha ugyanazt a permutációt alkalmazzuk a sorokra, mint az oszlopokra, tehát esetükben a mátrixon belüli közelség irreleváns.

#### Laplacian

A jelfeldolgozás és gráfelmélet határmezsgyéjén fekvő spektrális gráfanalízis területe azonban már évtizedekkel ezelőtt kidolgozott egy olyan, Laplace-mátrixnak (vagy csak Laplaciennek) nevezett reprezentációt[44], amely sajátértékeiben ír le strukturális információkat, így független a permutációtól.

**25. Definíció.** A  $G = (V, E)$  gráf **fokszám-mátrixaként** vagy **degree mátrixaként** ismerjük azt a  $D_G^{|V| \times |V|}$  négyzetes mátrixot, amelyre a diagonális elemeit a  $d_{i,i} = \text{degree}(v_i), i = 1, \dots, |V|$  képlet alapján kapjuk, minden egyéb eleme pedig 0.

**26. Definíció.** A  $G = (V, E)$  gráf **szimmetrikusan normalizált Laplace-mátrixa** az  $L = \sqrt{\text{inv}(D_G)} * A * \sqrt{\text{inv}(D_G)} = I - \sqrt{\text{inv}(D_G)} * A_G * \sqrt{\text{inv}(D_G)}$ , ahol  $D_G$  a  $G$  gráf fokszám-mátrixa,  $\text{inv}$  a mátrixinverz függvény, a gyökvonás elemenként értendő, a szorzás mátrixszorzás, az  $A_G$  az adjacencia mátrix és  $I$  a megfelelő méretű egységmátrix.

**Megjegyzés.** Egy másik Laplacian-variáns, a random walk (normalized) Laplacian is népszerű, azonban a továbbiakhoz nem szükséges a kettő közti különbséget mélyebben átlátni.

**Megjegyzés.** A Laplace-mátrix továbbra sem lesz független a gráf csúcsainak sorrendjétől, csak a rendezett sajátértékei.

### Számítási szabály

Bár a neurális hálók gráfjellegű adatokra való adaptációjára már korábban is voltak kísérletek[45, 46], napjainkban a – Welling és társa által kidolgozott – Laplace-mátrixon alapuló variáns[47] a leginkább elterjedt. Wellingék azt ismerték fel, hogy a megelőző réteg kimenete egy impulzusmátrixként is felfogható, amin – korábbi eredményekre hagyatkozva[48] – a Laplacian gráf alapú (spektrális) szűrőként funkcionál. Mivel ez a szűrő csak egy adott él környezetének megfelelő impulzusokat összegzi, ezért a számítás során nem az eredeti gráfot használják, hanem aszerint módosítják először, hogy minden csúcs egy hurokkel rendelkezzen. A neurális rétegekről szóló fejezetben látottaknak megfelelően tudjuk, hogy általában  $H^{(i+1)} = \sigma(H^{(i)} * W^{(i)} + B^{(i)})$  szabály segítségével kapható meg egy réteg kimenete. A Welling-féle gráf konvolúciós réteg (GCN) ezzel szemben a  $H^{(i+1)} = \sigma(\tilde{L} * H^{(i)} * W^{(i)})$  szabályt alkalmazza, amelyben  $\tilde{L}$  jelöli a hurokélekkel ellátott  $G$  gráf Laplace-mátrixát, azaz az  $\tilde{A} = A_G + I$  adjancia mátrix segítségével kapott Laplaciant. Mint látjuk, a biasok itt elhagyásra kerülnek, amit viszont azzal ellensúlyozunk, hogy  $\tilde{L}$  csak és kizárolag a lokális információkat szűri ki.

A GCN különösen alkalmas gráfcímkézési és klaszterezési problémák megoldására, akár alig néhány réteg használatával is.

#### 2.3.5. Hipergráf konvolúciós neurális hálózatok

Hipergráfok esetén már régóta vizsgált probléma, hogy hogyan lehet úgy általánosítani a gráfokon alkalmazott Laplaciant, hogy az minél több tulajdonságát megőrizze. Régebbi eredmények minden össze a hipergráfok speciális részosztályain voltak képesek ezt megvalósítani[49, 50], azonban egy 2015-ös cikkben[51] sikerült áttörést elérni, és egy általánosan alkalmazható definíciót bevezetni. A konstrukciós módszer bizonyos hiányosságait aztán 2018-ban Chan és társai tárták fel, illetve javították[52].

Ez a két eredmény lehetővé tette, hogy a GCN rétegekhez hasonló megoldásokat definálunk hipergráfok esetében. Először Feng és társai vetették fel, hogy a hipergráfokon értelmezett Laplace-mátrix alkalmazható GCN-szerű megoldásokra[53], azonban módszerükhez szükséges egy olyan gráf létrehozása, amelyben minden hiperél

klikként van reprezentálva. Nem sokkal később Yadati és társai több kisebb komplexitású metódust is felvetettek[54], amelyek – az eredményeik alapján – rövidebb tanítási időt igényelnek, de összemérhető vagy jobb eredményeket érnek el az általuk vizsgált problémán. Az általuk bemutatott módszerek (tudatosan) figyelmen kívül hagyták a feltárt Laplacian-konstruálási hibákat, azonban ez nem okozott problémát sem valós, sem szintetikus adatok esetében. Bandyopadhyay-ék egyszerűen a hipergráf line graph-ján futtatott GCN rétegekkel értek el ígéretes eredményeket[55]. A terület legújabb eredménye Tranék cikke[56], amelyben (általunk nem definiált) irányított hipergráfokra is kiterjesztik a módszert.

### Hipergráf Laplacian

Az alábbiakban áttekintjük a hipergráf Laplacian konstrukciós szabályát, ahogy azt az első, Chang-féle cikkben alkalmazták[51].

Adott  $H = (V, E)$  hipergráf és  $X \in \mathcal{R}^{|V|}$ , véletlenszámokat tartalmazó vektor.

1. minden  $e \in E$  hiperél esetén legyen  $\{i_e, j_e\} = argmax_{i,j \in X}(|X_i - X_j|)$ , a döntetlenek esetén véletlenszerű választással.
2. Hozzunk létre egy  $G_X = (V', E')$  irányítatlan, súlyozott gráfot, amelyben  $V' = V$  és  $E' = \{\{i_e, j_e\} | e \in E\}$ , továbbá  $w(\{i_e, j_e\}) = w(e)$ .  $G_X$  minden  $v$  csúcsához adjunk egy hurokéletet, amelyre  $w(v, v) = degree(v) - \sum_{e \in E: v \in \{i_e, j_e\}} w(e)$ .
3. Jelölje az  $A_X$  azt a mátrixot, amit  $G_X$ -ból, ha minden sort leosztunk az adott sornak megfelelő csúcs fokszámával.
4. Egyszerű gráfokhoz hasonlóan a szimmetrikusan normalizált hipergráf Laplaciant az  $L = (I - \sqrt(inv(D)) * A_X * \sqrt(inv(D)))$  képlet adja, azonban a konvolúciós réteg alkalmazása során  $A_X$  helyett az  $\tilde{A} = A_X + I$  mátrixot használjuk.

## 3. fejezet

# Saját eredmények

### 3.1. Vizsgált módszerek

#### 3.1.1. Euler-diagram reprezentáció

Láthattuk, hogy Euler-diagramok reprezentálhatók Bézier-görbékkel, körökkel, ellipszisekkel és poligonokkal is, azonban bizonyos esztétikai vagy szemantikai kritériumok betartása mellett (lásd well-formed kritériumok) nem minden hipergráf rajzolható ki.

Általános megoldás eléréséhez mindenkor egy olyan modell bevezetése szükséges, amely egyszerre képes kezelní az ismert esztétikai metrikákat, így például kompakt módon, egyetlen görbével reprezentálni egy halmazt, azonban mikor ez nem lehetséges, akkor – bár komplexebb diagramok használata mellett – továbbra is megfelel az elvárt szemantikai követelményeknek. Fogalmi szinten ennek természetesen megfelel az általánosított Euler-diagram definíció. Ennek a definíciónak a használatával viszont további kérdések merülnek fel. Mivel zárt görbék segítségével írtuk le a fogalmat, ezért kérdéses, hogy magukat a görbéköt hogyan ábrázoljuk a memóriában, illetve olyan reprezentációra lenne szükség, ami lehetőleg flexibilis, kevés paraméterrel leírható és tetszőleges számú régióra tud bontani egyetlen görbét, lehetőleg kis számítási igény mellett.

## Definíció

Az általam választott reprezentáció csúcsalapú, amelyben az egyes görbéket a hiperéleket alkotó csúcsok részhalmazainak konvex burka adja. Vegyük minden csúcs esetén egy háromelemű vektort, amely tartalmazza a csúcs  $x$  és  $y$  koordinátáit, illetve egy  $d$ -vel jelölt értéket, azaz egy  $X \in \mathcal{R}^{|V| \times 3}$  mátrixot. minden  $e$  hiperélhez rendeljünk egy  $G_e$  gráfot, amelynek csúcshalmaza megegyezik a hiperél csúcshalmazával, tetszőleges  $u, v, u \neq v$  csúcsa között pedig akkor fut él, ha  $distance(u, v) \leq min(u_d, v_d)$ , ahol a  $distance$  függvény tetszőleges távolságmetrika lehet (én az implementáció során az  $L^2$  normát használom). A  $G_e$  gráf összefüggő komponensei partionálják a hiperél elemeit. Az egyes partíciók – melyeket én a hiperél/görbe **szegmenseinek** fogok nevezni – konvex burkát véve a hiperél(et reprezentáló görbe) több zárt poligonra bomlik szét, így általánosított Euler-diagramot reprezentál. A reprezentáció előnye, hogy tetszőleges számú nem-üres komponensre bontható vele minden egyik él, illetve az egyes görbek az általuk tartalmazott csúcsok számával arányos komplexitással bírnak.

**Megjegyzés.** A továbbiakban  $Seg_e$ -vel jelölöm az  $e$  hiperél összes szegmensének halmazát.

Nem minden esetben előnyös a konvex burok alkalmazása, hiszen egy szegmensben tartalmazott egyetlen csúcs esetén nem jelenne meg az az információ, hogy melyik (ha bármelyik) hiperél tartalmazza azt. A két csúcs esetén előforduló hasonló helyzetben pedig a konvex burok egy egyenest adna, amelynek címkéje/színe – feltehetőleg – nem látható eléggé. Ebből az okból kifolyólag egy megkövetelt minimum távolságot,  $r$ -et is elvárunk a probléma definiálása során. Egy csúcs esetén konvex burok helyett az  $r/2$  sugarú kört, míg két csúcs esetén a két csúcs körül  $r/2$  sugarú köröket, és azok konvex burkát együttesen tekintjük a szegmenst tartalmazó görbéknek.

**Megjegyzés.** Nem garantált, hogy az elvárt minimális távolság teljesüljön a végső megoldásban, ekkor átfedő görbeket kapunk, amik szemantikai problémákhoz vezethetnek. Az egyetlen valós alternatíva, hogy  $r$ -et a konkrét csúcstávolságok minimumaként definiáljuk, azonban ekkor szintúgy problémákba ütközünk, ha több csúcs is egy pontba esik.

**Megjegyzés.** Az elfajuló esetet, mikor egy vagy két csúcsú szegmens egy nagyobb csúcsszámúból úgy alakul ki, hogy a csúcsok egy pontba esnek, nem kezelem külön. Bár ez valószínűleg előnyös lenne, azonban felvet olyan értelmezési kérdéseket, hogy az adott görbe hány csúcsot tartalmaz pontosan, mi a teendő, ha az egyik egy él egységi eleme és így tovább. Mivel ez az eset alapvetően kevéssé valószínű, ezért ezeknek a kérdéseknek a megválasztását inkább elkerültem, azonban egy későbbi kutatás célja lehet.

### Futásidő és memóriaigény

Figyeljük meg, hogy egy adott gráf tetszőleges csúcsának elhagyásával egy korábban összefüggő komponens előfordulhat, hogy két részre esik, azonban ez nem szükségszerű. Az előző okból kifolyólag minden hiperére külön kell létrehozzuk a  $G_e$  gráfot. Az egyes  $G_e$  gráfok akár minden  $e$  hiperél esetén tartalmazhatják az összes csúcsot, ezért feltehetjük, hogy a csúcsszáma megegyezik a hipergráfaval. A hiperélenkénti teljes gráf létrehozásának költsége  $\mathcal{O}(|E_H| * |V_H|^2)$ , ahol a  $H = (V_H, E_H)$  hipergráfot vizsgáljuk. minden  $G_e = (V_G, E_G)$  gráf esetén egy bejárást kell lefuttassunk, ami  $\mathcal{O}(|V_G| + |E_G|) = \mathcal{O}(|V_G| + |V_G|^2) = \mathcal{O}(|V_G|^2) = \mathcal{O}(|V_H|^2)$  futásidővel bír, azaz a teljes futásidő továbbra is  $\mathcal{O}(|E_H| * |V_H|^2)$ .

Mivel a csúcsonkénti  $x$  és  $y$  koordináták, illetve a  $d$  távolság alkotja a modellt, ezért a memóriaigény  $3 * |V| \in \mathcal{O}(|V|)$ .

### Megjegyzések

Csúcsok ábrázolása eredetileg nem része az Euler-diagramoknak, azonban a pozíciójuk tartalmazhat információt (esetleg használhatunk glyph-jellegű ikonokat is), így akár átértelmezhetik az ábra szemantikáját (például a létrejöttükkel engedélyezhetjük üres halmazok megjelenítését). Az általam használt módszer alkalmas csúcsok kezelésére, hiszen a modellben mindegyik konkrét pozícióval rendelkezik, viszont konkrétan a görbek határolóegyenésén is elhelyezkedhetnek (hiszen az ő konvex burkokat használjuk görbek leírására). Amennyiben a csúcsok megjelenítése is célnunk, akkor érdemes lehet utólagos lépésekkel alkalmazni a görbekre (vagy a széleken elhelyezkedő csúcsok pozícióra), de a csúcsok elrejtése is egy lehetséges megoldás.

A továbbiakban azt is érdemes lehet megvizsgálni, hogy nem csúcsonkénti, hanem globális vagy hiperélenkénti  $d$  értékek használatával mennyire romlik tanulási/optimalizálási idő, illetve az elérhető legjobb költségfüggvény.

### 3.1.2. Optimalizációs módszerek

A kísérletek során optimalizációs módszerek több kategóriáját hasonlítom össze; a particle swarm optimization (PSO) nevű módszert, illetve a genetikus algoritmus (GA) különböző variánsait. Bár – szigorúan véve – nem optimalizációs módszer, de itt fogom vizsgálni a költségfüggvény neurális hálókkal való közelítését is, amely célból hipergráf konvolúciós hálózatokat (HCNN) próbálok ki.

#### Particle swarm optimization

Az első algoritmus során van a legegyszerűbb dolgunk, mivel az eredeti – a 2.3.1 fejezet 2. számú pszeudokódjaként bemutatott – módszert egyáltalán nem módosítottam. Amennyiben az Euler-diagramot reprezentáló  $X \in \mathcal{R}^{|V| \times 3}$  mátrixot sorvagy oszlopfolytonosan adjuk meg az eljárásnak, viszont feltételezzük, hogy a költségfüggvény is ilyeneken operál, akkor az eredeti megoldás tökéletesen fedi a mi esetünket is.

#### Genetikus algoritmus

Emlékezhetünk a 2.3.1 fejezetből, hogy a GA leírása (lásd 1. pszeudokód) nem teljeskörűen kidolgozott. Bár az egyes lépések – mint szelekció, kereszteződés és mutáció –, meghatározhatók általánosan is, de gyakran inkább problémaspecifikus definíciót alkalmazunk. Mivel a modellben az egyes gének nem függetlenek egymástól – amennyiben egy csúcs  $x$  és  $y$  koordinátáját nincs értelme külön kezelní –, ezért én olyan módon határoztam meg az algoritmus szóban forgó lépéseit, amik ezt a tulajdonságot figyelembe veszik. Mint látni fogjuk a 3.1.4 fejezetben, a költségfüggvényt alkotó heurisztikák direkt úgy lettek kialakítva, hogy – amelyik esetében ez lehetséges – hiperélenként számítják a költséget, melyek aggregációjaként áll elő a globális költségfüggvény. Ez a számítási mód lehetővé teszi, hogy a GA során akár tetszőleges értékeket, akár hipercsúcsokat vagy hiperéléket tekintsünk a legkisebb

értelmes egységnek, amelyet örökölni lehetséges. Ennek megfelelően három különböző GA variánst hoztam létre, melyeket rendre naív GA, csúcsalapú GA és élalapú GA névvel fogok illetni.

A születést/halált modellező szelekciós függvényt és a szülők kiválasztását (melyek az erdeti leírásban együttesen *selectFitnessProportionally* néven, *population, fitnessValues, selectionRate* paraméterekkel szerepeltek) két lépésre bontottam.

---

### 3. Algoritmus Szelekció (saját szerkesztés)

---

**Function** *selection(population, fitnessValues, selectionRate)*

- 1: *selectionSize* =  $|population| * selectionRate$
  - 2: *worstIndividuals* = *selectionSize* db. legrosszabb értékkel bíró egyed (lecserejük új egyedekre)
  - 3: *population* = *population* \ *worstIndividuals*
  - 4: *population* = *population*  $\cup$  *generateFeasibleSolutions(selectionSize)*
  - 5: *fitnessValues* = *evaluateFitness(population)*
  - 6: *bestIndividuals* = *selectionSize* db. legjobb értékkel bíró egyed (kiválasztjuk a következő iterációban való megőrzésre)
  - 7: **return** *bestIndividuals*
- 

---

### 4. Algoritmus Szülők kiválasztása (saját szerkesztés)

---

**Function** *selectParents(population, fitnessValues, selectionRate)*

- 1:  $fitnessImpliedProbabilities = (fitnessValues - min(fitnessValues)) / sum(fitnessValues)$
  - 2: **for**  $i = 1 \dots |population| * selectionRate$  **do**
  - 3:     *parentIndices* = *fitnessImpliedProbabilities* alapú eloszlásból visszatevés nélkül kettő minta vétele
  - 4:     *parentsRow* = *population[parentIndices]*
  - 5:     *parents<sub>i</sub>* = *parentsRow*
  - 6: **end for**
  - 7: **return** *parents*
- 

A *recombine* (máshol gyakran *crossover* névvel illetett) függvény a kiválasztott szülőket kapja meg paraméterként, és kombinálja a génállományukat. Az itt tárgyal-

tak közül ez a legfontosabb függvény, mivel ez különbözteti meg a három variáns működését. A naív GA esetében a leszármazottak összeállítása mindenből áll, hogy egyenletes eloszlás szerint kiválasztjuk a gének felét, melyeket az első szü'lő génállokányából másolunk le, a többöt pedig a másodikéből. Csúcsalapú GA-nél mindenban törünk el ettől, hogy az  $x, y$  és  $d$  értékeket közösen tekintjük egy génnek, így ezek minden együttesen öröklődnek egy szülőtől (továbbra is fele-fele arányban).

Élalapú GA esetében természetesen az egyes egyedkhez nem elég egyetlen fitness értéket nyilvántartani, hanem minden egyed minden élre szükséges azt tárolni. Az élalapú megoldás esetén a *recombine* függvénynek ezek az élalapú fitness értékek is bemenetei, melyeket az *edgewiseFitness*  $\in \mathcal{R}^{|population| \times |E|}$  változó tartalmának tekintünk ( $H = (V, E)$  esetén), továbbá a függvény megkapja a hipergráf incidencia mátrixát is. A szóban forgó eljárás során továbbra is egy  $x, y, d$  hármast tekintünk génnek, így ezek csak együtt örökölhetők, azonban egy gén szülőjének kiválasztása a korábbiaktól eltérő módon történik. Az egyes csúcsokhoz minden külön-külön meghatározzuk, hogy az adott csúcsot tartalmazó éleknek mennyi az átlagos fitness értéke. Az öröklődés során kiválasztjuk azokat a géneket (azaz csúcsonkénti  $x, y, d$  értékhármasokat), amelyeknek ez az átlagolt értéke nagyobb az egyik rögzített (mondjuk első) szülőben. A kiválasztást úgy módosítjuk, hogy minden fele-fele arányban örököljünk a két szülőből. Ehhez a két átlagolt fitness érték különbségét vesszük, és azokkal bővíjtük/csökkentjük a kiválasztott szülőből átvett értékeket, amelyek a legkisebb eltérést mutatják. Így elveszíthetjük a legjobb értékek öröklését, azonban csak akkor, ha a másik szülő többi génjében nagyobb lenne a fitness értékben beállt veszteség.

A mutációt az új populáció alkalmazott normális eloszlású zajjal modellezük, amelynek az átlaga nulla, a felső értéke a *mutationRate* változó. Az én megoldásomban továbbá bevezetek egy *mutationPct* változót is, amely azt írja le, hogy az egyes paraméterekre milyen valószínűsséggel alkalmazzuk a mutációt.

## Hipergráf konvolúciós hálózat

A neurális hálózaton alapuló megoldás esetében fontos felismernünk, hogy nem egy direkt optimalizációs megoldásról beszélünk. Függvényapproximációs módszerként itt nem egyetlen hipergráf legjobb elrendezését szeretnénk megkapni, hanem

hipergráfok egy egész – azonos csúcs- és élszámú – családját szeretnénk leképezni (problémapéldányonként) különböző megoldásokra. Ez sajnos (nagyságrendekkel) lassabb tanulási idővel jár, és elkerülhetetlen a vizsgalandó hipergráfok köréből minél több legenerálása, amely alapján a hálózat betanítható. A hátrányok mellett a módszer természetesen előnyökkel is rendelkezik; a modellt elég egyszer betanítani, onnantól közvetlenül alkalmazható tetszőleges hipergráf Laplace-mátrixára.

A korábbiakban már bevezettük mind a hipergráf Laplaciant (2.3.5 fejezet), mind a gráf konvolúciós neurális hálózatokat (2.3.4 fejezet). Ahogy láttuk, több módszer is definiált hipergráf konvolúciós hálózatokra létrehozására. A kísérleteim során egy egyszerű variánst választottam ezek közül, amely megegyezik a GCN számításával, azonban a Laplace-mátrixot lecseréljük a hipergráf Laplacianjára. A módszer nem csak intuíción alapuló kísérlet, Yadatiék cikkében[54] a FastHyperGCN nagy hasonlóságot mutat vele, minden össze én nem alkalmazok mediátorokat (máshogy súlyozom a Laplacian éleit).

A tanítás természetesen nem SGD alapú, hiszen a költségfüggvénytől nem várjuk el, hogy deriválható legyen, így a naív GA eljárást választottam a neurális háló betanítására. Mivel a gráf és hipergráf konvolúciós hálózatokkal foglalkozó cikkek jellemzően kettő rétegű architektúrát alkalmaznak[47, 54], ezért én sem fogok mély hálózatokkal kísérletezni. Az egyes rétegekhez *relu* ( $f(x) = \max(0, x)$ ), *sigmoid* ( $f(x) = e^x / (e^x + 1)$ ) vagy *softmax* ( $f_i(x) = e^x / \sum_{i=1}^n (e^x)$ ) aktiválási függvényt használok (ahol emlékezzünk, hogy az aktiválási függvény minden neuron kimeneti értékére külön-külön számítódik), azzal a kikötéssel, hogy az utolsó réteg nem lehet *relu* (mivel nem szorítaná az értékeket egy előre meghatározható, korlátos intervallumba). Neurális hálók tanítása során kevésbé látható át, hogy az egyes paraméterek hogyan befolyásolják a végeredményt, azonban közismert, hogy a paraméterek nagyságrendjének növekedése negatívan befolyásolja a hálózat tanulásának sebességét[57]. Ebből az okból kifolyólag a paramétereket kis értékekkel ( $mean = 0, \sigma^2 \in (0, 1], \theta = (mean, var)$  paraméterű normális eloszlással) inicializálom, ami után a hálózat sekélysége biztosítja, hogy *relu* esetén se kapunk nagy értékeket (ezért közvetlen regularizációt nem alkalmaztam).

### 3.1.3. Inicializációs módszerek

A bemutatott algoritmusok során többször előfordul, hogy megengedett megoldásokat kell generálunk. Tekintve, hogy ezeknek a minősége nagyban befolyásolja a futás során elérhetőkét, ezért különösen fontos kérdés, hogy kezdeti értékeket hogyan számítjuk ki. Ebből a célból a dolgozatban több módszert is vizsgálok a megoldások inicializására, melyeket ebben a fejezetben részletezlek.

#### Egyenletes eloszlás alapú inicializálás

A legegyszerűbb – és az optimalizációs algoritmusok eredeti leírásában általában alkalmazott – módszer, hogy a modell minden dimenziójához megadunk egy alsó és felső értéket, majd az inicializálás során a kettő közötti egyenletes eloszlás szerint választunk új értéket. A mi esetünkben az  $x$  és  $y$  koordináták felső határát az alkalmazott képméret adja meg, míg a  $d$  esetében akkor kapjuk a legnagyobb értelmes számot, ha a két átellenes csúcs távolságát nézzük, amely Pitagorasz-tétellel megkapható. Az alsó korlátok természetesen minden esetben nullával egyeznek meg.

#### Élenkénti inicializálás

Ennek az inicializálási módszernek a során először az élek pozícióit határozzuk meg (az egyenletes eloszlás alapú inicializálás szerint), majd az egyes csúcsokat az őket tartalmazó élek átlagaként értelmezzük. A  $d$  értékek ettől függetlenül, csúcsonként számítódnak ki. A mögöttes feltételezés, hogy így olyan klaszterek alakulnak ki, amely egy-egy csúcs összes élét tartalmazzák.

#### Erőalapú gráfelrendezés szerinti inicializás

Egy hipergráfot leírhatunk gráfszerű struktúrákkal is (pl. a 2.2.1 fejezetben bevezetett bipartite incidence structure vagy szuperduális segítségével). A hiperélek klikkekre való cserélése első ránézésre ilyennek tűnhet, azonban figyeljük meg, hogy alkalmazásuk során elveszik az az információ, hogy pontosan melyik hiperélből származik a két csúcs közötti él a gráfban. Amennyiben továbbra is az előző módszernél bevezetett megfontolásból indulunk ki (azaz a szomszédok valamiféle átlagolásával elérhető klasztereket szeretnénk kialakítani), akkor azonban az így elvesző információ egyáltalán nem is szükséges. Tekintsük azt a fizikai szimulációt, amelyben a

nem szomszédos csúcsok taszítják, a szomszédosak pedig vonzzák egymást. Ekkor a rendszer egyensúlyi helyzetei felfoghatók ilyen átlagolásként is. Az így leírt módszer pontosan a force-directed gráfkitrajzolási módszer (lásd 2.2.2) egy speciális esetét írja le. Az általam használt konkrét eljárás a Fruchterman-Reingold módszer[8], amelyet (a cikkből átemelve) alább láthatunk.

---

### 5. Algoritmus Fruchterman-Reingold (Fruchterman alapján[8])

---

**Function** layout(*Canvas* = (*width*, *height*), *G* = (*V*, *E*), *iterations*)

```

1:  $k = \sqrt{width * height / |V|}$ 
2:  $t = \sqrt{width^2 + height^2}$            lehetséges maximum (a cikkben nem definiált)
3: for  $i = 1 \dots iterations$  do
4:   for  $v \in V$  do                  taszítóerők kiszámítása
5:      $v.disp = 0$ 
6:     for  $u \in V$  do
7:       if  $\{u, v\} \notin E$  then
8:          $v.disp = 0$ 
9:        $\Delta = v.pos - u.pos$ 
10:       $v.disp = v.disp + (\Delta / |\Delta| * k^2 / |\Delta|)$ 
11:      end if
12:    end for
13:  end for
14:  for  $v, u \in E$  do                vonzóerők kiszámítása
15:     $\Delta = v.pos - u.pos$ 
16:     $v.disp = v.disp - (\Delta / |\Delta| * |\Delta|^2 / k)$ 
17:     $u.disp = u.disp + (\Delta / |\Delta| * |\Delta|^2 / k)$ 
18:  end for
19:  for  $v \in V$  do
20:     $\Delta = v.pos + (v.disp / |v.disp| * \min(v.disp, t))$        $t$  szerint maximáljuk az
      elmozdulást
21:     $v.pos.x = \min(width / 2), \max(-width / 2, v.pos.x)$  nem engedjük túlfutni a
      kép szélén
22:     $v.pos.y = \min(height / 2), \max(-height / 2, v.pos.y)$ 
23:  end for

```

---

---

24:	$t = cool(t)$	<i>csökkentjük a maximális lehetséges elmozdulást</i>
25:	<b>end for</b>	

---

Az algoritmus során kezdetben egyenletes eloszlás szerinti véletlen  $v.pos$  értékeket tételezünk fel.

### Súlyozott inicializálás

Mikor több, mint egy megoldást inicializálunk, akkor megtehetjük, hogy az új elemek felét az egyik, felét egy másik inicializálási módszerrel definiáljuk. Ezt a módszert általánosíthatjuk tetszőleges számú inicializációs módszerre (amelyek esetleg csak a paraméterezeitükben térnek el), illetve fele-fele bontás helyett tetszőleges súlyozásra is.

### Távolságértékek statisztikai inicializálása

Minden korábbi módszer esetén a csúcsok pozicionálására koncentráltunk, a  $d$  értékek háttérbe szorultak. Ezt ellensúlyozandó az egyes módszerek során lehetőség van ezeket különböző módokon számítani. A fönt bevezetett véletlenszerű módszeren kívül – a csúcsok koordinátáinak beállítása után – minden adott  $v$  csúcs  $d$  értékét beállíthatjuk a legközelebbi, legtávolabbi vagy átlagos szomszéd távolsága alapján, ahol minden csúcsot szomszédnak tekintünk, amellyel  $v$  közös hiperélben szerepel.

#### 3.1.4. Heurisztikák

Az egyik legfontosabb tulajdonsága az általam használt Euler-diagram modellnek, hogy nem rigid szabályokon alapszik (például well-formed vagy well-matched kritériumok teljesítésén), hanem súlyozható (és költségfüggvényként alkalmazható) mérőszámokon. A kapott ábrával szembeni elvárásaink természetesen sokrétűek lehetnek, azonban az irodalom áttekintése során láttuk, hogy egyes tulajdonságok több szerző szerint is segítik a végeredmény könnyebb értelmezését, emberi feldolgozását. A szabályalapú megoldásokat kiváltandó, az általam legfontosabbnak ítélt tulajdonságokat igyekeztem különböző heurisztikus mérőszámok segítségével leírni. A létrehozott heurisztikus módszerek, illetve egyáltalán az általuk leírt tulajdonságok köre is vitathatatlanul némileg önkényesen lettek meghatározva, azonban a dol-

gozat egyik fő célja pontosan annak kiderítése, hogy mennyire alkalmasak a vizsgált módszerek arra, hogy tetszőleges esztétikára szabva tudjunk hipergráfokat kirajzolni. Mivel kizárolag a költségfüggvényen alapszik a modell tanítása, ezért az magában kell foglalja a szemantikus információkat is, azaz hogy az ábrázolt csúcsok és görbék elhelyezkedése megfelel-e az Euler-diagram definíciójának. Bár ezutóbbi tulajdon-ság jóval fontosabbnak tűnhet, mint az esztétikai megfontolások, azonban könnyen előfordulhat például, hogy a hiperélek egymáshoz viszonyított relatív mérete több információt hordoz a szemlélő számára, mint amit néhány csúcs rossz hiperélbe való helyezésével elveszít.

A heurisztikák által visszaadott mérőszámok persze nem alkalmasak arra, hogy önmagukban költségfüggvényként használjuk őket, hiszen csak egy-egy aspektusát vizsgálják az adott ábrának, ezért inkább veszteségfüggvényként, a költségfüggvény építőelemeként gondolhatunk rájuk. Ahogy már korábban is említettem, a heurisztikák direkt úgy lettek kialakítva, hogy ahol lehetséges, hiperélenként is értelmezve (és tárolva) legyenek, a többi esetben pedig csak globálisan. minden mérőszám a  $[0, 1]$  intervallumból veszi fel az értékeit, ahol a 0 a legjobb, 1 a legrosszabb lehetséges érték. Ezzel a módszerrel kerültem el, hogy a különböző skálákból adódó értelmezés nehézség ellehetetlenítse az eredmények összehasonlítását. Igyekeztem a mérőszámokat úgy kialakítani, hogy ne csak a szélsőértékek legyenek felvethetők, hanem több diszkrét ugrás is legyen a két végpont között. Az ugrások jobban közelítenek egy folytonos függvényt, amely többletinformációt szolgáltat az optimalizációs algoritmusok számára, hogy a paraméterek adott irányú elmozdulása pozitívan befolyásolja-e a költségfüggvényt. Kísérleteim során azonban közvetlenül is vizsgálni fogom, hogy csak a szélsőértékek használata (azaz gyakorlatilag szabályok elvárása) ténylegen negatívan befolyásolja-e az algoritmusokat. A költségfüggvény kiszámításához hiperélenként súlyozva összeadjuk az arra alkalmas heurisztikák eredményét, majd ezek átlagához hozzádvha a csak globálisan értelmezettek eredményét, megkapjuk az aggregált, végső értéket. Ezek a súlyok a felhasználó által meghatározhatók, így a költségfüggvény egy paraméteres függvény.

**Megjegyzés.** *Operálhatnának azzal a feltételezéssel a heurisztikák, hogy az ábra tere  $[0, 1] \times [0, 1]$ -en értelmezett, majd a végfelhasználó átméretezhetné ezt a saját igényeinek megfelelően. Mivel az ábrák szélessége és magassága jellemzően nem 1 : 1*

arányú, ezért ezzel olyan torzítást is bevezetnénk, amely nem jelenik meg a modellben. Ezt elkerülendő a költségfüggvény, így a heurisztikák is paraméterként megkapják, illetve hozzáférnek az ábra szélesség/magasság attribútumához.

**Megjegyzés.** A szemfűles olvasónak feltűnhet, hogy a genetikus algoritmust végig úgy tárgyaltam, hogy maximalizálási feladaton operál, azonban a költségfüggvényt egyféléképp definiálom itt. Ez a tényleges megvalósítás során nem okozott problémát, mivel pontosan ebből a célból magát a genetikus algoritmust minimalizációs eszköz-ként alkalmaztam.

## Szemantikus heurisztikák

Elnevezés	Leírás
<i>Hamis pozitív tartalmazás</i>	Egy adott $v$ hipercsúcstól elvárjuk, hogy a neki megfelelő pont az ábrán ne szerepeljen egyetlen olyan hiperélt reprezentáló görbüben sem, amely hiperél nem tartalmazza őt. (Hiperélekre értelmezett mérőszám.)
<i>Hamis negatív tartalmazás</i>	Egy adott $v$ hipercsúcstól elvárjuk, hogy a neki megfelelő pont az ábrán szerepeljen minden olyan hiperélt reprezentáló görbüben, amely hiperél tartalmazza őt. (Konstans szám.)

A *hamis pozitív tartalmazást* úgy vizsgálom, hogy minden  $e$  hiperél minden  $Seg_j \in Seg_e$  szegmense esetén egy hibásan tartalmazott csúcs  $dist(v, Seg_j) / (\min(width_{Seg_j}/2, height_{Seg_j}/2) * |\{v|v \in V, v \notin Seg_j\}| * |Seg_e|)$  mennyiséggel járul hozzá a metrika értékéhez, ahol  $dist(v, Seg_j)$  jelöli a tartalmazott csúcs távolságát a poligon széleitől,  $width_{Seg_j}$  és  $height_{Seg_j}$  pedig a szegmenst ábrázoló poligon befoglaló négyzetének megfelelő paramétereit (pixelben). A nevező utolsó két tagja normalizálja a szegmenseken és nem tartalmazott csúcsokon való iterációt, míg az első tag a távolságot. Mivel a konvex burok egy belső pontjának a szélektől való legnagyobb lehetséges távolságának meghatározása nem triviális, ezért azt a befoglaló négyzettel közelítjük, amely felülről becsüli azt. Ez természetesen azt jelenti, hogy nem feltétlenül érhető el a (legrosszabb) 1 veszteségfüggvény-érték, viszont az továbbra is a  $[0, 1]$  intervallumból fog kikerülni.

*Hamis negatív tartalmazás* a mi esetünkben nem fordulhat elő, hiszen a csúcsok konvex burkait használjuk, amely szükségszerűen tartalmazza az őt kifeszítő pontokat. Ebből kifolyólag ez a mérőszám csak a teljesség kedvéért (illetve az esetleges modellcsere miatt) került be, azonban esetünkben konstans 0 értéket kap.

**Megjegyzés.** *Konvex poligonok esetén egy pont tartalmazásának a vizsgálata  $\mathcal{O}(\log(n))$  időben elvégzhető[58].*

### Esztétikai heurisztikák

Elnevezés	Leírás
<i>Körszerűség</i>	A szimmetriát, speciálisan a körhöz való hasonlóságot értékeli. (Hiperélekre értelmezett mérőszám.)
<i>Szegmensek száma</i>	Bünteti a szegmensek számának növelését. (Hiperélekre értelmezett mérőszám.)
<i>Szegmensek mérete</i>	Előnyben részesíti az azonos méretű szegmenseket. (Hiperélekre értelmezett mérőszám.)
<i>Minimum távolság fenntartása</i>	A csúcsok közti távolságok minimumát igyekszik a célérték fölé tornászni. (Globálisan értelmezett mérőszám.)
<i>Minimum távolság előfordulása</i>	Csökkenti a célszám alatti csúcstartások előfordulásának számát. (Globálisan értelmezett mérőszám.)
<i>Területarányosság</i>	Egy adott hiperélet reprezentáló görbék területének az ábra teljes méretével vett arányát közelíti az általa tartalmazott csúcsok számának az összes csúccsal vett arányához. (Hiperélekre értelmezett mérőszám.)
<i>Szegmensmetszetek száma</i>	Minimalizálja a görbék metszeteit. (Globálisan értelmezett mérőszám.)
<i>Invalid szegmens-metszetek száma</i>	Minimalizálja a metszettel nem rendelkező éléket reprezentáló görbék metszeteit. (Globálisan értelmezett mérőszám.)

<i>Legközelebbi szomszéd élszomszéd</i>	Támogatja, hogy egy csúcs legközelebbi szomszédja közös élből származzon. (Hiperélekre értelmezett mérőszám.)
<i>Fragmentálatatlanság</i>	Bünteti az egyetlen csúcsból álló szegmensek számát. (Hiperélekre értelmezett mérőszám.)

Számos Euler-diagramokat vizsgáló cikk koncentrál kifejezetten körök és ellipszisek által leírt halmazrendszerekre. Az egyszerű kezelhetőségen felül ez felveti annak a lehetőségét, hogy a szimmetrikus görbék ergonomiai előnyökkel járnak, amit Blake és társai meg is erősítettek[33]. A *körszerűséget* (circularity) vizsgáló heurisztika a megjelenő poligonokat igyekszik közelíteni a körökhöz. Bár ez korlátozottan lehetséges (gondoljuk csak a három vagy négy csúccsal reprezentált polinomokra), azonban – a képfeldolgozás területén – már használatos a poligonoknak pontosan ezt a tulajdonosságát kifejező mérőszám. Alaktényezőnek (shape factor) nevezzük az  $F = 4*\pi*A/P^2$  számot, ahol  $A$  egy zárt poligon területét,  $P$  a kerületét írja le. Az általam használt mérőszámoknak megfelelően a  $[0, 1]$  intervallumból veszi fel az értékeit, viszont 1 jelentvén, hogy tökéletes kör, míg attól távolodva egyre nagyobb az eltérés a „kör-ségtől”. Mivel én pont fordítva vezettem be a skálát, ezért annak megfordítását az  $1 - F$  képlettel kapjuk. Egy teljes hiperél értékét a  $\sum_{Seg_j \in Seg_e} (1 - F(Seg_j)) / |Seg_e|$  normalizált összeg adja meg.

Egyik legfőbb céлом volt, hogy általánosan alkalmazható Euler-diagram modellt dolgozzak ki, azonban a hiperélek több szegmensre bonthatósága növeli az ábra komplexitását is. Támogatandó az egyszerűbb(en értelmezhető) kirajzolásokat, szükséges a szegmensekre bontás büntetése. A *szegmensek számát* a hiperélenkénti  $1 - 1/|Seg_e|$  mérték használatával igyekszem leszorítani. Ahogy a szegmensek száma a végtelenbe tart, úgy ez az érték 1-hez konvergál, míg egyetlen görbéből álló él esetén nullára értékelődik ki.

Nem tűnik az sem előnyösnek, hogy a szegmensek mérete nagyban ingadozzon. Amennyiben ezt támogatnánk, akkor megnövekedhet a kevés csúcsot tartalmazó részhalmazok száma, ami megnehezítheti az egy halmazhoz tartozó görbék áttekintését. Hogy igyekezzünk elkerülni a kis csúcossal szegmenseket, minden  $e$  hiperére kiszámítjuk a  $\sum_{Seg_j \in Seg_e} ((|V_{Seg_j}| / |V_e| - 1 / |Seg_e|) / |Seg_e|)$  értéket. A belső

( $||V_{Seg_j}|/|V_e| - 1/|Seg_e||$ ) szám megadja, hogy egy adott szegmens csúcsainak száma mennyire tér el a célként meghatározott egyenletes eloszlástól, míg a külső osztótényező a szumma normalizálásának feladatát látja el.

Egymáshoz túl közel elhelyezett csúcsok összecsúszhatnak a kirajzolás során, akár teljesen el is fedhetik egymást. Elkerülendő ennek az előfordulását, a kirajzoló algoritmus ismeretebén meghatározhatunk egy olyan távolságot, amelyet az optimalizációs algoritmus aztán igyekszik minden csúcspár esetében biztosítani. Erre a célról hoztam létre a *minimum távolság fenntartása* heurisztikát, amelyet  $1 - \min(min_{actual}/min_{target}, 1)$ -ként határoztam meg, ahol  $min_{actual}$  a csúcsok között ténylegesen előforduló legkisebb távolság, míg  $min_{target}$  a bemenetként megadott célszám. A korábbiakkal ellentétben ez egy globálisan, nem pedig hiperélenként értelmezett heurisztika. Amennyiben  $min_{actual}$  nagyobb a célnál, akkor  $1 - 1$ -ként értékelődik ki a függvény, azaz minimalizálási feladatban optimálisként. Ellenkező esetben egy lineáris függvényt kapunk 1 és 0 között, ahogy a tényleges minimum közelíti a célt. A globális heurisztikák nagyobb súlyval számítódnak a végeredménybe (hiszen nem átlagolódnak), azonban az optimalizációs algoritmus számára kevesebb információt szolgáltatnak, hogy mely irányba lehet érdemes folytatni a keresést.

Felmerül a kérdés, hogy mikor nagyszámú csúcspár távolsága van a kitűzött cél alatt, akkor hogyan ismerje fel az optimalizációs algoritmus, hogy ezek számát csökkenteni előnyös. *Minimum távolság előfordulásaként* hivatkozok arra a globálisan értelmezett arányszámra, hogy az összes lehetséges páron belül hány kisebb, mint  $min_{target}$ , azaz  $|\{(u, v) | u, v \in V, u \neq v, dist(\{u, v\}) < min_{target}\}|/(|V| * (|V| - 1)/2)$ .

Az Euler-diagramok ergonómiáját áttekintő 2.2.4 fejezetben láthattuk, hogy a méretarányosság egy gyakran vizsgált tulajdonság a szakirodalomban. A fogalom azt írja le, hogy minden  $Region_i$  régió esetén az  $Area_{Region_i} / \sum_{Region_j \in Regions} (Area_{Region_j})$  arány azonos a – régiókon értelmezett  $\omega$  súlyfüggvény alapján megkapott –  $\omega(Region_i) / \sum_{Region_j \in Regions} (\omega(Region_j))$  aránnyal. A  $\omega$  függvény gyakran a tartalmazott csúcsok számát írja le. Mivel a régiók száma exponenciális lehet az élek számához képest, ezért csak nagyon kis példákon alkalmazzák közvetlenül ezt a fogalmat, módszerről-módszerre eltérő mértékszámokat alkalmazva. Az általam kialakított heurisztika során – amelyet *területarányosságnak* fogok nevezni – a polinomiális futásidő és a méretarányossági tulaj-

donság legalább részleges megtartása is a szemem előtt lebegett. Ennek megfelelően régiók helyett minden  $e$  hiperél esetén csak az azt alkotó szegmenseket vizsgálom, a  $\sum_{Seg_i \in Seg_e} (|Area_{Seg_i}/Area_{Canvas} - |V_{Seg_i}|/|V|)/|Seg_e|)$  képlet alapján. A korábbiaknak megfelelően itt is egy célértéktől (a csúcsaránytól) való eltérést igyekszünk minimalizálni, egy normalizációs faktorral ( $|Seg_e|$ ) leosztva. A méretarányossággal ellentétben itt a teljes ábra méretéhez viszonyítom a görbe területét, hogy ne fordulhasson elő, hogy az összes szegmens területarányosan, de az ábrának csak nagyon kis részterületén helyezkedjen el.

Ismert, hogy az egymást metsző görbek körül megnevezül az ábra értelmezése (gondolunk csak gráfok síkbarajzolhatóságára). Érdemes ezért alacsonyan tartani a görbemetszetek számát, amihez a globálisan kiértékelt *szegmensmetszetek számát* minimalizáljuk. Természetesen minden olyan él tudja metszeni egymást, ami nem ugyanazon élből származik (azok közös konvex burkolatnak egyébként). Az elérhető metszetek száma  $(\sum_{e \in E} (|Seg_e|)) * (\sum_{e \in E} (|Seg_e|) - 1)/2 - \sum_{e \in E} (|Seg_e| * (|Seg_e| - 1)/2)$ , a különböző élekben lévő szegmensek metszeteinek leszámlálása során ezzel osztunk le.

*Invalid szegmensmetszeteknek* fogom nevezni azt az esetet, mikor két görbe úgy metszi egymást, hogy az általuk reprezentált hiperéleknek nincs közös eleme. Ezeknek a számának alacsonyan tartásával csökkenthetjük az üres metszetek létrejöttét az ábrán. A számítás hasonlóan történik az előző esethez, azonban itt csak akkor számlálunk egy metszéspontot, ha az invalid, illetve a normalizációs faktort  $(\sum_{e \in E} (|Seg_e|)) * (\sum_{e \in E} (|Seg_e|) - 1)/2 - \sum_{e \in E} (|Seg_e| * (|Seg_e| - 1)/2) - \sum_{e, e' \in E, e \neq e', e \cap e' \neq \emptyset} (|Seg_e| * |Seg_{e'}|)$ -nak választjuk, hiszen szomszédos élek szegmensei között sem fordulhat elő számlált metszet.

Élmetszeteket, így speciálisan a szegmensek metszeteit kiszámítani különösen lassú művelet, melyet hasznos lehet inkább közelíteni. Az invalid szegmensmetszetek közvetlen leszámlálása helyett egy alternatív heurisztikát is bevezettem, amely azt vizsgálja, hogy egy csúcs legközelebbi szomszédja szerepel-e vele közös hiperélben. A feltételezésem, hogy ez megnehezíti az invalid metszetek létrejöttét, mivel nem hagyja, hogy nem szomszédos csúcsok ékelődjenek egy szegmens csúcsai közé. Ez természetesen csak egy hipotézis, amit a méréseim során külön vizsgálni fogok.

Mikor általános megoldást keresünk hipergráfok egy teljes osztályára (azaz spe-

ciálisan a konvolúciós módszer esetében), akkor a triviális megoldás a szemantikai és számos ergonómiai heurisztika minimalizálására, hogy minden csúcsot egyben teljes szegmensnek is választunk (amit a  $d$  értékek alacsonyan tartásával érhetünk el). Ezt a problémát elkerülendő bevezettem még egy mérőszámot a *fragmentálatlanság*-ot. A fragmentálatlanság hiperélenként  $|\{Seg_i | Seg_i \in Seg_e : |Seg_i| = 1, |Seg_e| \neq 1\}| / |Seg_e|$ -ként van definiálva, azaz a nem egyetlen csúcsból álló hiperélek egyetlen csúcsból álló szegmenseinek arányaként.

## Összehasonlítás a szakirodalommal

Érdemes megvizsgálni, hogy a modell tulajdonságai, illetve a fenti heurisztikák mennyire felelnek meg a szakirodalomban megjelenő ergonómiai kritériumoknak.

Először a 2.2.4 fejezetben tárgyalt well-formedness, kikötéseit veszem végig. A WFC1 kritérium (a görbék nem metszik önmagukat) mindenképp teljesül, hiszen az egyes görbék konvexek, míg a modellt direkt úgy alkottuk meg, hogy a WFC6 (nem rendelkezik két görbe azonos címkével) ne legyen elvárás (így az nem teljesül). Részben függ a kirajzolástól a WFC2 (nincs görbéknek közös határolószakasza) és WFC4 (két görbe metszi egymást, ha érintkezik). Ezekben az esetekben ugyanis a konvex burkok megnövelhetők a csúcsok közti minimális távolság felével, ami szemantikailag nem változtatja meg az ábrát (hiszen mindegyik görbe ugyanazokat a csúcsokat fogja tartalmazni), viszont minden közös határolószakaszt metszetté alakít. Ezzel szemben sem a modell, sem az alkalmazott heurisztikák nem fedik le a WFC3 (nincs pont, ahol három él érintkezik) kritériumot. A WFC5 ( minden zóna egyetlen minimális régióból áll, azaz részhalmazok nem üres metszete pontosan egy összefüggő síkpartícióval van reprezentálva) indirekt módon le van fedve a szegmensek számának csökkentésével, illetve a szegmensek metszeteinek minimalizálásával, azonban direkt módon nem optimalizáljuk. Jegyezzük meg, hogy bár széleskörűen alkalmazott a well-formedness kritériumrendszer, azonban az empirikus kísérletek nem támasztották alá a hasznosságát.

Nem mondható ez el a – ugyanott tárgyalt – well-matched követelményekről, amelyről azt állapították meg az ergonómiai kísérletek, hogy segíti a halmazábrák emberi értelmezését. Itt kissé nehezebb dolgunk van, mivel a csúcsok megjelenítésével nem tisztán Euler-diagram alapú reprezentációt kapunk, ami megnehezíti a

kritériumok értelmezését. A WMP1 tulajdonság kimondja, hogy nem jelenik meg olyan részhalmazt reprezentáló görbe, amely üres. Az eddig részletezett modellben ez előfordulhat – bár az invalid szegmensmetszetek minimalizálása pont ezt igyekszik elkerülni –, azonban ezek nem fognak csúcsokat tartalmazni. A WMP2 és WMP4 (a halmazok relációi megfelelnek a görbék átfedésének) ugyanettől a problémától szenved; maguk a görbék önmagukban nem feltétlenül teljesítik, de a csúcsok figyelembevételevel már igen. A WMP3, előírja, hogy a WMP2-n felül, a WFC5 is teljesüljön. Láttuk az előző paragrafusban, hogy egyes heurisztikák célozzák ugyan ezt a tulajdonságot, azonban nem régióként vizsgálva. Ennek oka, hogy a minimális régiók bejárása exponenciálissá tenné a futásidőt, amit igyekeztem elkerülni. Összességében elmondhatjuk, hogy minden WMP kritériumot céloz heurisztika, továbbá, amennyiben a csúcsok megjelenítése a nem üres halmazokban az emberi percepciót nem nehezíti, akkor tökéletes megfelelés is elérhető. Az utóbbi feltevés természetesen csak ergonómiai kísérletek során bizonyítható.

Az előző kettő esztétikai kritériumrendszeren kívül még a méretarányosságot emeltem ki az irodalmi áttekintés során, azonban annak kapcsolatát a heurisztikákkal már részleteztem a területarányosságot leíró paragrafusban.

### 3.1.5. Összefoglalás

Az előző fejezetekben a megoldási módszer részleteit fejtettem ki, azonban a jobb átláthatóság érdekében úgy gondolom, hasznos egy helyen is áttekinteni, hogy azok hogyan állnak össze egésszé.

A teljes folyamat során egy adott hipergráfhoz generálunk általanosított Euler-diagramot. A diagram a hipercsúcsokhoz rendelt  $x, y$  koordinátákkal és egy-egy  $d$  küszöbértékkel íródik le. A  $d$  értékek alapján az egyes hiperéléket ábrázoló görbék több részre eshetnek szét (részletekért lásd a 3.1.1 fejezetet), melyek mindegyike az őket alkotó csúcsok konvex burkaként lesz kirajzolva.

Optimalizációs módszereket alkalmazok (3.1.2), hogy hipergráfokhoz szemantikai és ergonómiai szempontból is megfelelő diagramot társítsunk. Ezek a módszerek magukban foglalják a genetikus algoritmus variánsait, particle swarm optimizationt és egy függvényapproximációt (genetikus algoritmus segítségével tanított hipergráf konvolúciós hálózat) módszert is. Költségfüggvényként tulajdonságok és heurisztikai

kák alapján kialakított mérőszámok (felhasználó által) súlyozott átlagát használom (3.1.4). A költségfüggvény által lefedett tulajdonságok igyekeznek lefedni a szakirodalom által hatékonynak találtakat.

Elkerülhetetlen, hogy az optimalizálási eljárás során új, véletlenszerű megoldásokat legyen szükséges előállítanunk, melyek a további megoldások kiindulópontjaként funkcionálnak. Különböző inicializációs módszerekkel (3.1.3) igyekszem javítani az ilyen, kezdeti megoldások minőségét.

Alkalmazás során a felhasználó nagyfokú szabadsággal élhet a módszerek testreszabása terén. Természetesen ki kell válassza az általa használni kívánt inicializációs és optimalizációs módszereket, utóbbi méghozzá saját paraméterekkel rendelkezik. A költségfüggvény egyes összetevőinek súlyait szintén igény szerint lehet változtatni.

Szigorúan véve egyedül a hipergráf kéne leírja a problémát, azonban a költségfüggvény úgy lett kialakítva, hogy az ábra elvárt mérete és a csúcsok távolságára kitűzött alsó korlát is szükséges bemenet.

## 3.2. Mérések és következtetések

Közismert, hogy az optimalizációs módszerek csak meghatározott függvényosztályokra, speciális kritériumok teljesítése mellett garantálható, hogy az optimumhoz konvergálnak. Ebben a dolgozatban az első perctől egy általános problémára igyekeztem megoldást találni, amely során a költségfüggvényben szakadások, ugrások találhatók, a paramétertér mérete pedig nem elhanyagolható. Ezt ugyan igyekeztem ellensúlyozni a megválasztott heurisztikákkal, azonban mindenkiépp kétségesi teszi a módszerek eredményességét. Különösképp igaz ez, mivel a megoldási módszerekről csak részben mondható el, hogy organikusan (korábbi eredményekre alapozva) fejlődtek volna. A szakirodalomban mindenkiössze egyetlen hasonló kísérlet lelhető fel[23], amely során azonban annyira különböző (az általánosság rovására menő, de jobb tulajdonságokkal rendelkező) Euler-diagram modellt és költségfüggvényt alkalmaztak, hogy kevéssé tudtam építeni rá.

A gyakorlati alkalmazhatóság vizsgálatának céljából egy referenciaprogramot is készítettem. A továbbiakban különböző – a referenciaprogramra épülő – szoftverkísérlet mentén elemzem, hogy az egyes algoritmusok alkalmasak-e valós problé-

mapeldányok megoldására, illetve milyen paraméterek mellett alkalmazhatók vagy hatékonyak.

### 3.2.1. Mérési környezet

Mielőtt áttekintjük az eredményeket, minden képp szükséges pár szót ejtenem magáról a programról, illetve a kísérletekről is. Implementáció során – a módszerek nagy száma, illetve a kutatómunka kiterjedtsége okán – sajnos már nem jutott idő a futásidő extenzív optimalizációjára. Az egyes algoritmusok polinomidejű futásidejét nem veszítettem szem elől, azonban valószínűsíthető, hogy további fejlesztésekkel szignifikáns sebességnövekedés is elérhető lenne. Érdemes ezért észben tartani, hogy a látott futásidők nem feltétlenül reprezentatívak egymáshoz viszonyítva. Ennek ellenére úgy gondolom, hogy így is hasznos információt hordozhatnak a futásidő felső korlátjaként.

A mérések során a bemenet fixen  $1080 \times 720$  pixel méretű ábrát és véletlenszerűen generált hipergráfot foglal magában. A generált hipergráf megadott csúcs- és élszám-mal rendelkezik, illetve minden csúcs meghatározott valószínűséggel tartalmazott az egyes hiperélékben (azzal a kikötéssel, hogy minden hiperél tartalmaz legalább egy csúcsot). Ezeket a paramétereket minden kísérlet előtt megadom, továbbá egy-azon kísérlet során pontosan ugyanazt a generált hipergráfot alkalmazom az összes módszer esetén. Egy olyan generált hipergráfot, amely  $N$  csúccsal,  $M$  hiperéllel és  $C\%$  tartalmazási valószínűséggel bír,  $H_{N,M,C}$  módon fogok jelölni. Szintúgy jelezni fogom, hogy melyik inicializációs módszert, illetve milyen paraméterezésű optimalizációs módszert használom. A további alfejezetekben az általam legfontosabbnak vélt aspektusokat vizsgálom, minden ilyen előtt egy rövid leírás a mérés témaáról, illetve céljáról. Mivel a hipergráf konvolúciós hálózat egészben más természetű módszer, mint a többi, ezért nem éreztem jól összehasonlíthatónak velük, így egy külön neki dedikált alfejezetben tárgyalom.

Ahol nem jelölöm külön, ott az alábbi súlyokat használom az egyes heurisztikákhoz:

Heurisztika	Érték
<i>Hamis pozitív tartalmazás</i>	10.000

<i>Hamis negatív tartalmazás</i>	10.000
<i>Körszerűség</i>	10
<i>Szegmensek száma</i>	1000
<i>Szegmensek mérete</i>	100
<i>Minimum távolság fenntartása</i>	1000
<i>Minimum távolság előfordulása</i>	10
<i>Területarányosság</i>	10
<i>Szegmensmetszetek száma</i>	10
<i>Invalid szegmensmetszetek száma</i>	1000
<i>Legközelebbi szomszéd élszomszéd</i>	0
<i>Fragmentálatlanság</i>	1000

Ezek az értékek úgy lettek kiválasztva, hogy a szemantikára vett – általam vélt – befolyásuk alapján súlyoztam a heurisztikákat. Természetesen más súlyozás is ugyanennyire megalapozott lehet, azonban túl sok paraméter játszik szerepet az optimalizáció során ahhoz, hogy minden kombinációt vizsgáljak. Ugyanez természetesen fennáll az algoritmusok paraméterei esetén is. Jegyezzük meg, hogy a szegmensmetszetek alternatívájának szántuk annak mérését, hogy egy csúcs legközelebbi szomszédja élszomszédja-e, ezért azt egyelőre teljesen kihagyjuk. Szintén konstans 0 értékkel fog szerepelni a költségfüggvényben a hamis negatív tartalmazás is, hiszen az mindig teljesül. Ennek ellenére hasonló súlyú szemantikai információnak ítélem, mint a hamis pozitív tartalmazást, így egy esetleges modellváltásnál a fönti értéket ajánlanám használni.

Legtöbb esetben azt a kérdést vizsgálom, hogy melyik módszer milyen paraméterek mellett milyen mértékben képes minimalizálni a költségfüggvényt adott iterációszám alatt. Ezen esetekben négy mérőszámot alkalmazok; a költségfüggvény értékét, a referenciaimplementáció által igényelt futásidőt, illetve a tanulóalgoritmus által vizsgált megoldáspopuláció elemeinek átlagát és szórását. Az első kettő gyakorlati haszna magától értetődő, míg a második kettő a tanulás minőségét hivatott reprezentálni. Amennyiben az utóbbiak nem csökkennek konzisztensen, akkor feltehető ugyanis, hogy a tanulás során csak véletlenszerűen bukanunk kisebb értékű pozíciókra. Ezzel szemben az átlag folyamatos csökkenése azt mutatja, hogy a kereső

algoritmus minden számítási kapacitását egyre jobb minőségű lokális minimumhe lyek köré fókuszálja.

Az ábrán megjelenő jó néhány tulajdonság konkrétan számszerűsíthető is, azonban ezek a költségfüggvényben arányosítva, a  $[0, 1]$  intervallumba szorítva jelennek meg. A konkrét mérési eredményeken kívül több ilyen tulajdonság eredeti leszámlálását is nyomonkövetem, hogy a heurisztikák eredményességére következtetni lehessen. Ezek a tulajdonságok magukban foglalják a szegmensenkénti hamis pozitív tartalmazások számát, a szegmensmetszetek számát, a nem szomszédos élek szegmensmetszeteinek számát, a meghatározott minimum küszöbértéknél közelebbi csúcspárok számát, a szegmensek számát, illetve az olyan szegmensek számát, amely annak ellenére tartalmaz egyetlen csúcsot, hogy maga a hiperél többől áll. Figyeljük meg, hogy a szegmensenkénti hamis pozitív tartalmazás maximum lehetséges értéke akár  $(|V| - 1) * |E|$  is lehet, mikor egyazon csúcs szerepel csak minden hiperélben, de az összes többit is fedi az általa kialakított szegmens. (Egy élen belüli szegmensek nem metszhetik egymást, úgyhogy egy csúcs legfeljebb  $|E|$  esetben lehet fals pozitív.)

### 3.2.2. Érzékenység a probléma paramétereire

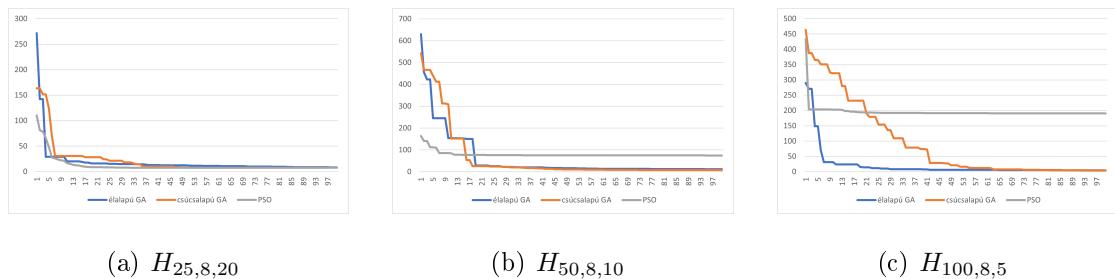
Értelemszerűen fontos kérdés, hogy a különböző módszerek legfeljebb mekkora hipergráfot képesek kezelni, a paramétertér növelésével hogyan romlik akár a teljesítményük, akár a futásidéjük. A kísérlet során különböző méretű és sűrűségű hipergráfokon futtatom ugyanazon optimalizációs algoritmusokat.

Inicializáció	Algoritmus	Algoritmus paraméterek
Egyenletes eloszlás	PSO	$S = 100$ , $w = c_1 = c_2 = 0.5$
Egyenletes eloszlás	csúcsalapú GA	$S = 100$ , $selectionRate = 0.2$ , $mutationPct = 0.3$ , $mutationRate = 3$
Egyenletes eloszlás	élalapú GA	$S = 100$ , $selectionRate = 0.2$ , $mutationPct = 0.3$ , $mutationRate = 3$

Méretfüggetlenül 100 iterációs lépést végzek a genetikus algoritmusok esetében, és 200-at a PSO-éban, amely futásidő alapján körülbelül kétszer olyan gyorsnak bizonyult. Diagramok esetén – az összehasonlíthatóság érdekében – az utóbbinak csak minden második értékét ábrázolom.

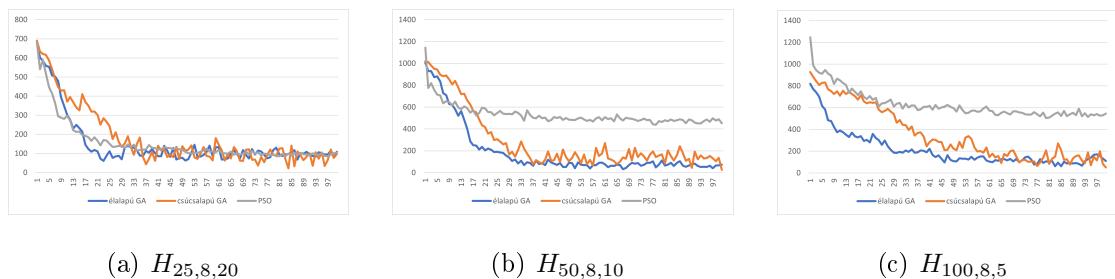
### Csúcsám növelése

A csúcsszám változtatásának hatását az algoritmusokra a  $H_{25,8,20}$ ,  $H_{50,8,10}$  és  $H_{100,8,5}$  hipergráfikon vizsgáltam. Fontos kiemelni, hogy a tartalmazási valószínűséget a csúcsszámmal arányosan csökkentettem, így az egyes hiperélek megközelítőleg azonos számú csúcsot tartalmaznak minden kísérlet során.

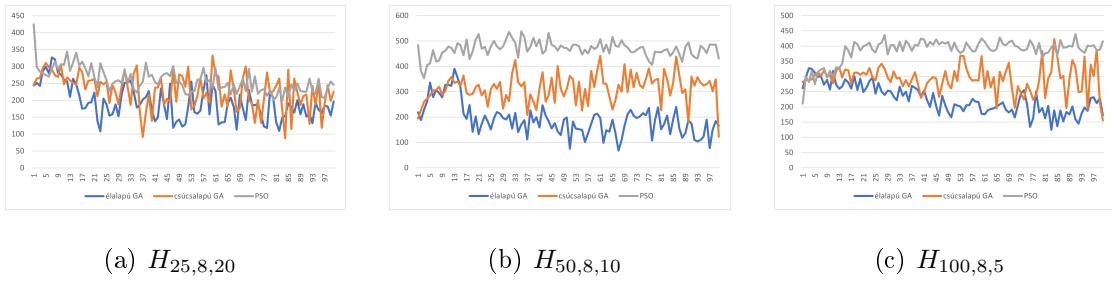


3.1. ábra. Csúcsszám hatása a költségfüggvényre

Úgy tűnik, hogy a a probléma ezen paraméterének megváltoztatása nagyban eltérően hat az egyes algoritmusokra. Míg az élalapú GA invariánsnak tűnik a csúcsszámra, addig a csúcsalapú GA esetében lassabb konvergenciához vezet, de végül mindenketten hasonló minőségű ábrákat generálnak. Ezzel szemben a PSO teljesítményét abszolút negatívan befolyásolja, ha további csúcsokat adunk a problémához.



3.2. ábra. Csúcsszám hatása a megoldáspopuláció költségfüggvényének átlagára

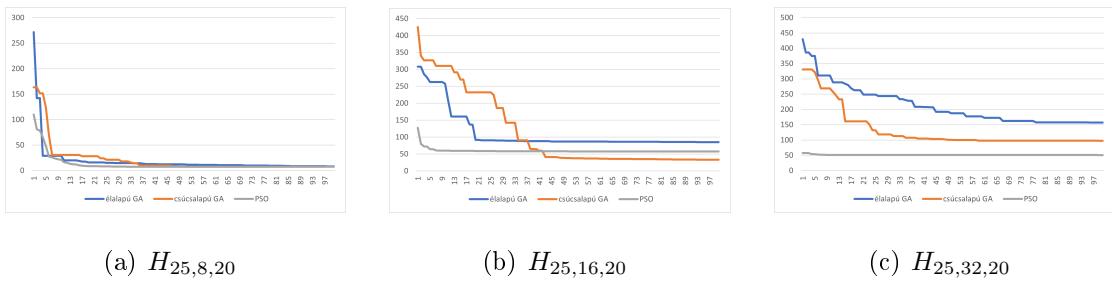


3.3. ábra. Csúcsszám hatása a megoldáspopuláció költségfüggvényének szórására

Nem meglepő a korábbiak alapján, hogy az optimalizáció során átlagosan a kettő GA algoritmus megoldáspopulációja hasonlóan jó eredményt ér el, míg a PSO-ról ez már nem mondható el. A szórást vizsgálva azt figyelhetjük meg, hogy a PSO során kapott költségfüggvények jóval nagyobb kilengéseket mutatnak, de a csúcsalapú GA is kevésbé stabil, mint az élalapú.

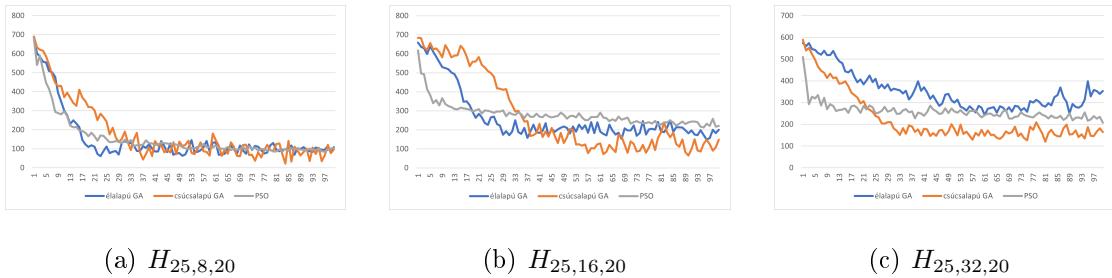
### Élszám növelése

Élszámok esetén  $H_{25,8,20}$ -at,  $H_{25,16,20}$ -at és  $H_{25,32,20}$ -at vettet górcső alá.

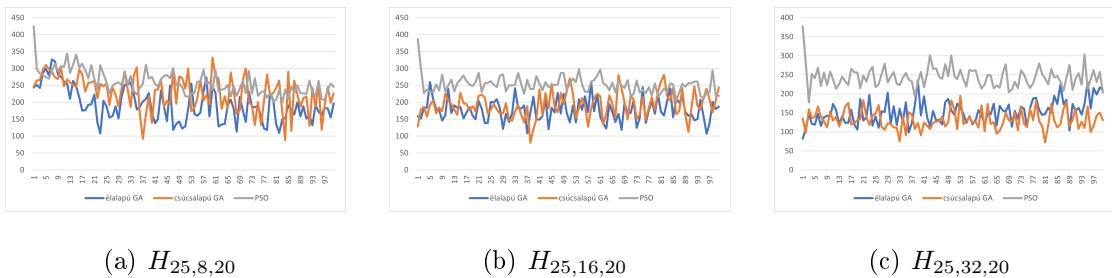


3.4. ábra. Élszám hatása a költségfüggvényre

Ebben az esetben a csúcsszámokkal pont ellentétes konklúzióra juthatunk, a PSO-t érinti legkevésbé az élszám növelése, az élalapú GA-t pedig a legnagyobb mértékben. Érdemes megemlíteni, hogy míg a csúcsszám változtatásával az elérhető költségfüggvény nagyságrendje nem változott, ez már nem áll meg az élszám esetén, ahol még a legjobb eredményt adó módszer teljesítménye is észrevehetően romlik. Meglepő módon átlagosan nem a PSO operál a minimális költségfüggvényű (és szórású) populációval, hanem a csúcsalapú GA.



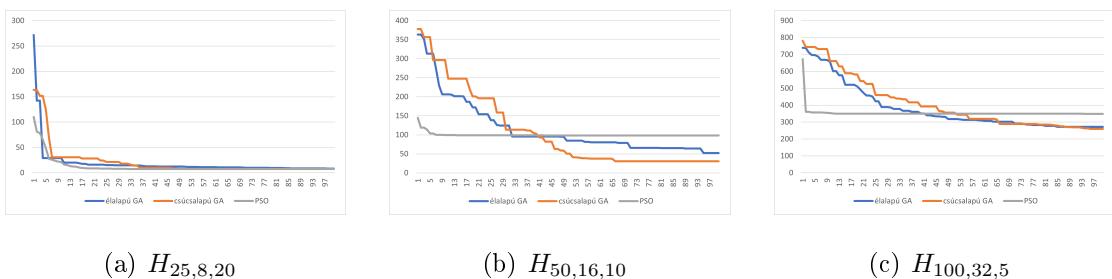
3.5. ábra. Élszám hatása a megoldáspopuláció költségfüggvényének átlagára



3.6. ábra. Élszám hatása a megoldáspopuláció költségfüggvényének szórására

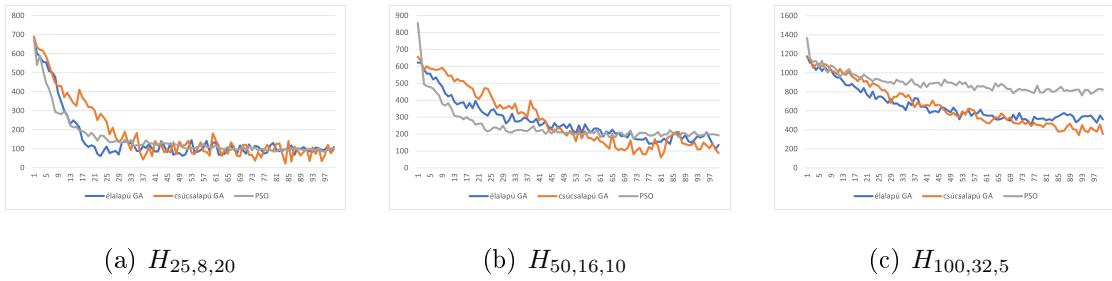
### Csúcs- és élszám együttes növelése

Mivel az előző két mérés során – a csúcsszám és az élszám növelésével – eltérő módszerek kerültek ki győztesként, ezért felmerül a kérdés, hogy mi történik, mikor minden kettőt egyszerre növeljük. Ehhez a korábbi mérések kombinációjával előálló  $H_{25,8,20}$ ,  $H_{50,16,10}$  és  $H_{100,32,5}$  hipergráfokat használom fel.



3.7. ábra. Csúcs- és élszám együttes szám hatása a költségfüggvényre

Jól látható, hogy ekkor minden módszer teljesítménye romlik, azonban, még a két GA hasonló ütemben konvergálni látszik, a PSO az optimalizáció elején megakad ennél a komplexitásnál.

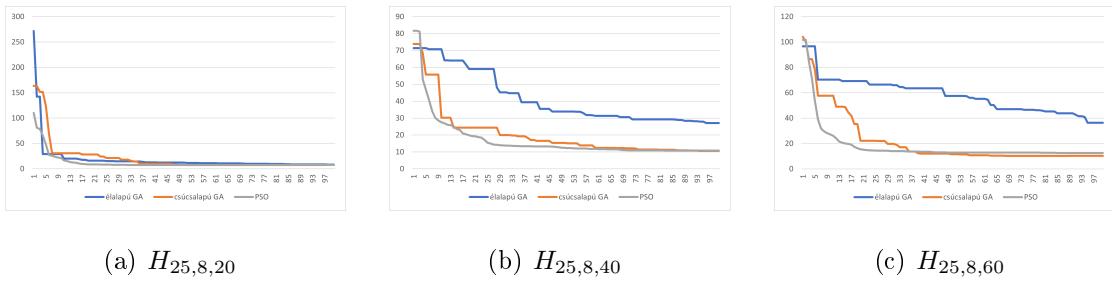


3.8. ábra. Csúcs- és élszám együttes hatása a megoldáspopuláció költségfüggvényének átlagára

Az átlagértéket vizsgálva is azt állapíthatjuk meg, hogy a PSO a  $H_{100,32,20}$  esetében már jóval elmarad a másik két módszer mögött. Érdekes összehasonlítani a GA módszereket a kizárolag éleket változtató kísérlettel; 50 csúcs esetén ugyanis hasonlóan teljesítenek, mint 25-nél, de a legnagyobb példán már feltűnően rosszabbul. Kérdéses, hogy ez a komplexitás növekedése miatt van vagy a véletlen hipergráfok harmadik paramétere miatt.

### Sűrűség növelése

Eddig a véletlen hipergráfok minden paraméterét vizsgáltuk, kivéve az élek tartalmazási valószínűségét, amelyet a hipergráf sűrűségeként is értelmezhetünk. A következő kísérletben a  $H_{25,8,20}$ ,  $H_{25,8,40}$  és  $H_{25,8,60}$  hipergráfokon keresztül ezek hatását láthatjuk.



3.9. ábra. Sűrűség hatása a költségfüggvényre

Bár a csúcsalapú GA és a PSO hasonlóan teljesít ennek a paraméternek a változatása során, az élalapú GA teljesítményét láthatóan nagyban rontja. Elképzelhető, hogy ekkora sűrűségek mellett már negatív hatása van annak, hogy az új megoldások minden két szülőtől fele-fele arányban öröklik a géneket.

### 3.2.3. Algoritmusok hatékony paraméterezésére

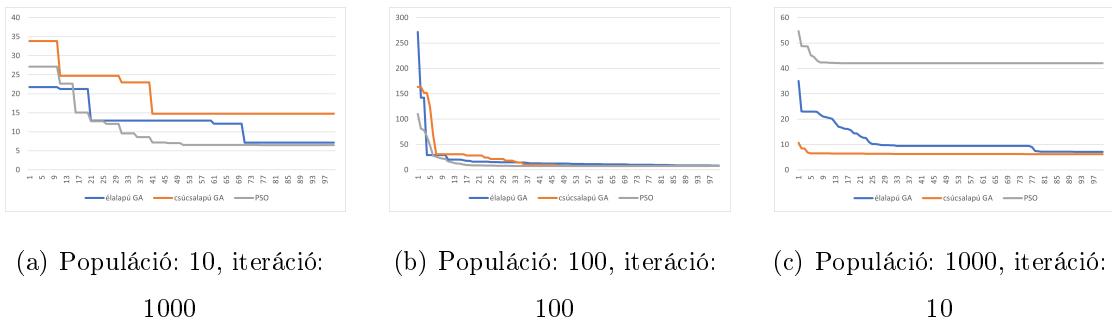
Az optimalizációs algoritmusok számára legmegfelelőbb hiperparamétereket  $H_{25,8,20}$ -on futtatott kereséssel igyekeztem kijelölni.

Genetikus algoritmusok esetében a  $selectionRate \in \{0.1, 0.3, 0.4\}$ ,  $mutationPct \in \{0, 0.3, 0.6, 1\}$  és  $mutationRate \in \{3, 10, 50\}$  értékek mellett futtattam az algoritmusokat. A kombinációk nagy száma miatt itt nem közlöm az összes kapott költségfüggvényt, csak a levont következtetéseket. Az eredmények között nincs nagy különbség, mind a  $[6.15, 8.6]$  intervallumba esik, jellemzően ennek is az alsó-középső tartományába. Élalapú GA esetén minden az 50-es  $mutationRate$  érték, minden a 0.4-es  $selectionRate$  negatívan befolyásolja a teljesítményt. A csúcsalapú GA esetén nincsenek kiugróan negatív eredmények, azonban 0.2-es  $selectionRate$  mellett gyorsabban konvergál. A legjobb eredményt, 6.16-ot a 0.2, 0.6, 10 paraméterezésű csúcsalapú GA adta, azonban a közeli eredmények miatt nem jelenthető ki, hogy tényleg kiemelkedő lenne.

Hasonlóan jártam el a PSO esetében is, ahol minden hiperparaméter a  $\{0.1, 0.5, 1, 2\}$  értékeket vehette fel. A kapott eredmények még szorosabbak lettek, mint a genetikus algoritmus esetében, jellemzően a  $[6.35, 8.7]$  intervallum alján elhelyezkedő eredményekkel. Az egyetlen egyértelmű negatív hatást az fejtette ki, mikor a  $c_1$  paraméter legalább 0.9-cel nagyobb volt a  $w$  paraméternél, ami nem meglepő tekintve, hogy az utóbbi paraméter határozza meg, hogy a részecskek mennyire preferálják az korábban látott globális optimum pozíójának közelítését.

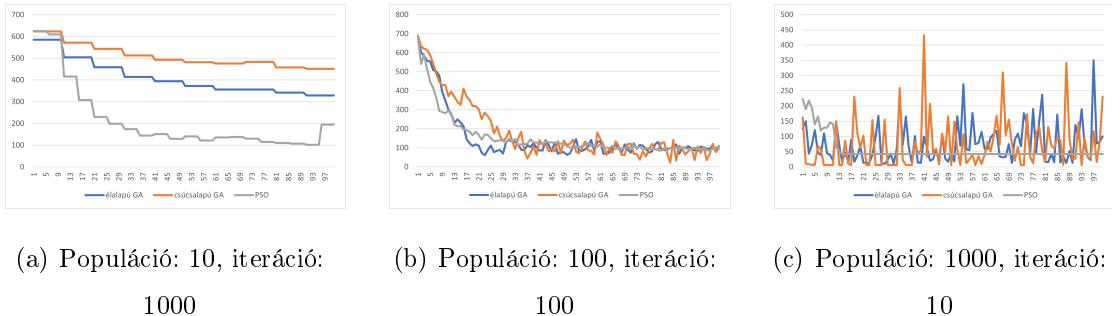
### Iterációk száma kontra megoldáspopuláció mérete

Feltehető, hogy az optimalizációs algoritmus legköltségesebb lépése a költségfüggvény kiértékelése. Amennyiben az iterációk és a megoldáspopuláció méretének szorzatát konstans értéken tartjuk, akkor kiértékelések száma nem változik, azonban elképzelhető, hogy valamely érték növelése előnyösebb a konvergencia szempontjából. Az alábbiakban  $H_{25,8,20}$ -on, 10000 kiértékelés mellett vizsgálom, hogy érdemes-e valamely paramétert előnyben részesíteni.



3.10. ábra. Költségfüggvény azonos számú kiértékelés mellett

Alacsony iterációs szám mellett egyedül a csúcsalapú GA nem ért el hasonlóan jó eredményt, az – eredetileg alkalmazott – 100-100 bontású futtatásához, azonban a költségfüggvények átlagai minden algoritmus esetében jóval elmaradtak attól. A skála másik szélén, 1000 iteráció mellett a PSO teljesen elbukott, amennyiben a futásának az elején konvergált egy adott – nem túl jó – ponthoz. A másik két algoritmus jobban teljesített, azonban az átlagok itt sem meggyőzőek, elég nagy ugrásokkal operálnak.



3.11. ábra. Költségfüggvények átlaga azonos számú kiértékelés mellett

Összességében levonható az a következtetés, hogy az iterációs számot érdemes a populációval arányosnak megtartani.

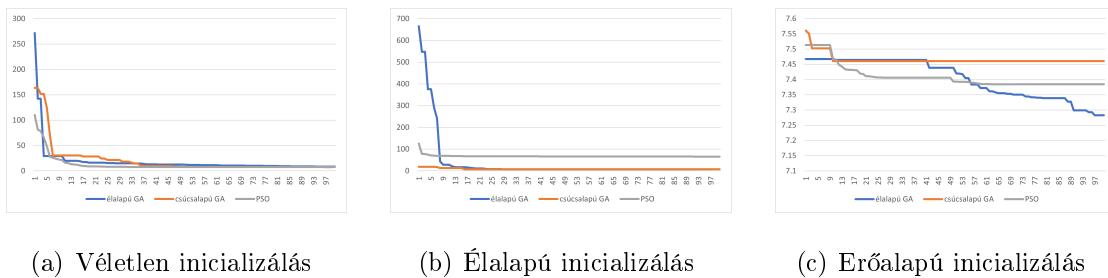
#### 3.2.4. Előnyös inicializálási módszerek

Optimalizációs algoritmusok során létrehozandó új megoldások generálására három különböző módszert vezettem be; az egyenletes eloszláson, éleken, illetve erőalapú gráf fel rendezésén alapulót. Felmerül a kérdés, hogy melyik milyen gyors konverenciához vezet, továbbá milyen költségfüggvény érhető el a használatukkal. Mint

a 3.1.3 alfejezetben említettem, a konkrét módszerek főként a csúcsonkénti  $x$  és  $y$  koordináták pozicionálására fókusznak, a  $d$  értékek háttérbe szorulnak. Ennek el-lensúlyozására a  $d$  értékek meghatározása véletlenszerűen, de az adott csúcs élszom-szédaiból távolságstatisztikai alapján is meghatározhatók. A különböző módszereket a  $H_{25,8,20}$ -on nyújtott teljesítményük alapján elemzem.

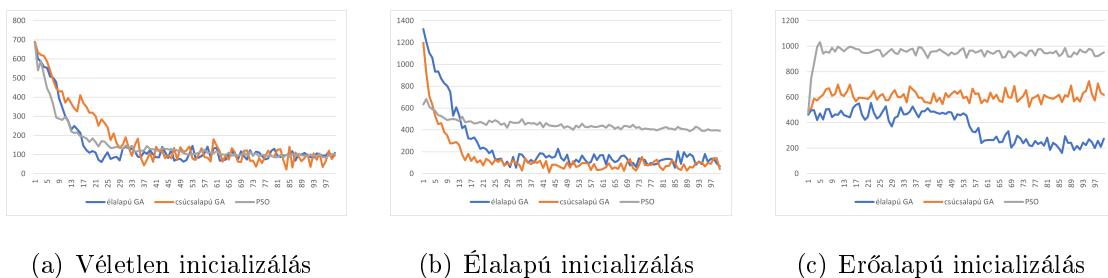
### Véletlenszerű $d$ értékek

Az első esetben, véletlenszerű  $d$  értékek használatával az figyelhető meg, hogy a különböző módszerek végül minden hasonlóan jó költségfüggvényt képesek elérni. Az egyedüli kivételt a PSO/élalapú inicializálás képezi, amely egy nagyságrenddel elmarad ettől. Érdemes megfigyelni, hogy a konvergencia sokkal gyorsabb az erőalapú módszer esetén, ahol már az első iteráció során közel kerülünk a futás során optimumhoz, azonban egyedül az élalapú GA tűnik képesnek további konvergenciára.



3.12. ábra. Inicializálás véletlen  $d$  értékkal - költségfüggvény

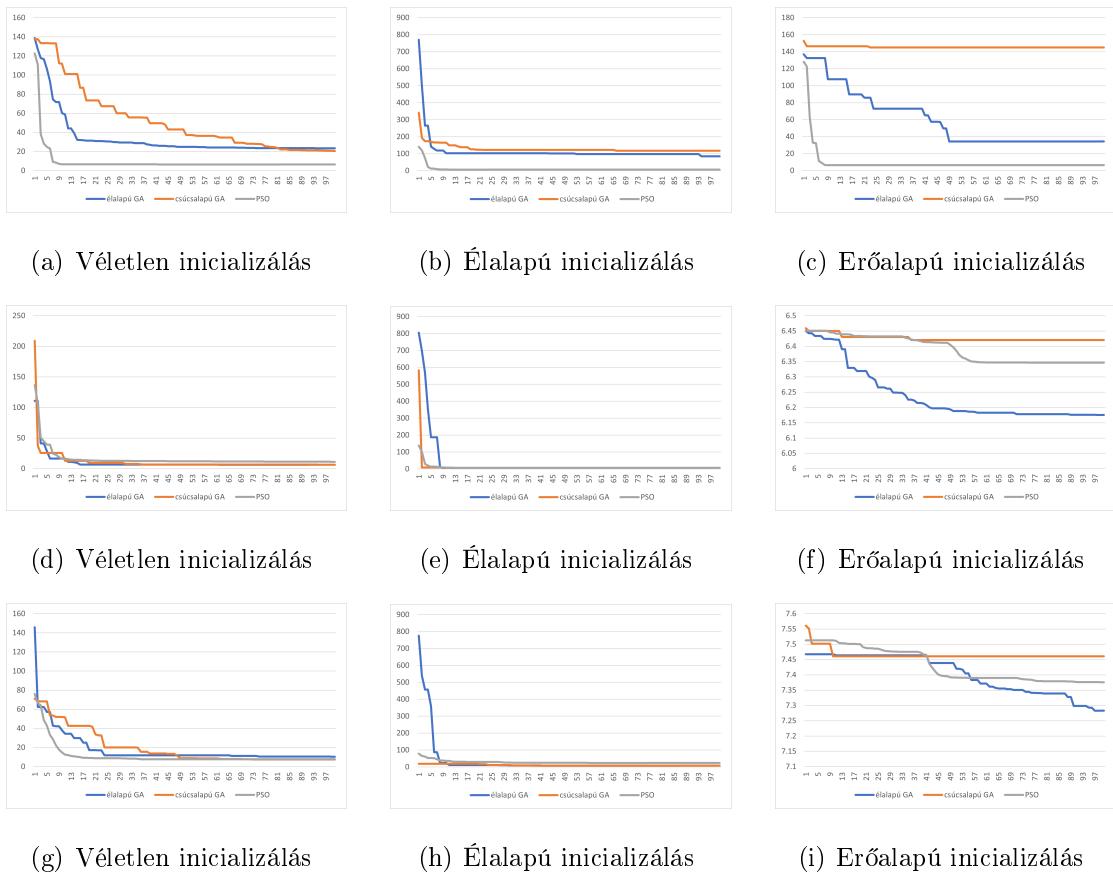
Megerősíti a korábbi megfigyeléseket a megoldáspopuláció átlagainak vizsgálata is, amely során azt láthatjuk, hogy a PSO átlaga elmarad a többi minden az élalapú, minden az erőalapú megoldás esetén, míg az utóbbinál egyedül az élalapú GA tud javítani a kezdeti értékén.



3.13. ábra. Inicializálás véletlen  $d$  értékkal - költségfüggvények átlaga

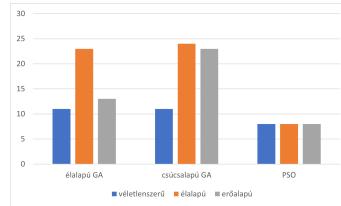
### Statisztikák a $d$ paraméteren

Fontos kiemelni, hogy a vizsgált hipergráf esetén az optimális kirajzolás élenként egy szegmenst használ (hiszen 8 halmazig minden hipergráf síkbarajzolható), így a maximum érték kiválasztása egyben a legjobb is kell legyen. Mivel ez több halmaz esetén nem feltétlenül igaz, ezért kérdéses, hogy a nagyobb számú szegmens kialakításához szükséges egyéb stratégiákat ellen tudják-e a súlyozni az optimalizációs módszerek.



3.14. ábra. Inicializálás, rendre min, max és átlagos  $d$  értékekkel - költségfüggvény

Az eredmények alapján ekkora hipergráfnál minden esetben körülbelül azonos eredményt érnek el az algoritmusok, a minimum statisztika kivételével. Ezutóbbi nem befolyásolja a PSO hatásfokát azonban a két GA-ét nagyban rontja. Véletlenszerű inicializálás esetén a PSO eredménye is romlani látszik (6 körüli költségfüggvény helyett 11-et ér el), azonban a szegmensek számának vizsgálatával megállapíthatjuk, hogy ez az inicializálási módszertől függetlenül történt (hiszen az továbbra is az elérhető minimum).

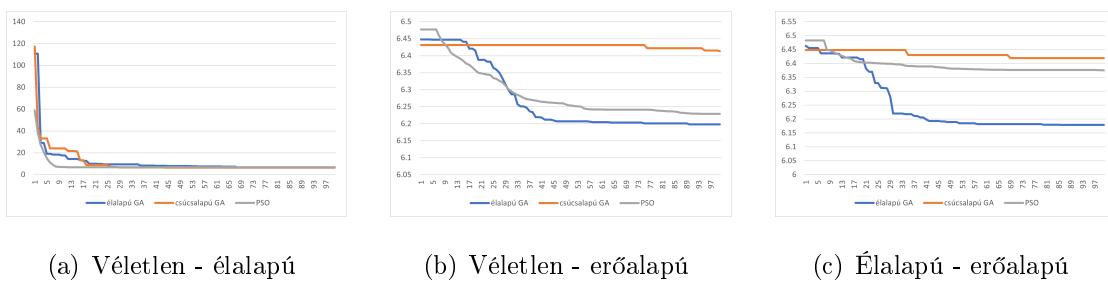


3.15. ábra. Szegmensek száma véletlenszerű, élalapú és erőalapú inicializálás esetén, min statisztikával

Természetesen tudjuk, hogy a szegmensek számának közvetlen hatása van a költségfüggvényre, azonban a mérések szerint ezt az algoritmusok nem is tudják ellen-súlyozni; a kiugróan rosszul teljesítő megoldások (azaz a minimum statisztikák segítségével inicializált GA-k) egyben az egyedüliek, amelyek több, mint 8 szegmenst használnak a végső elrendezés során.

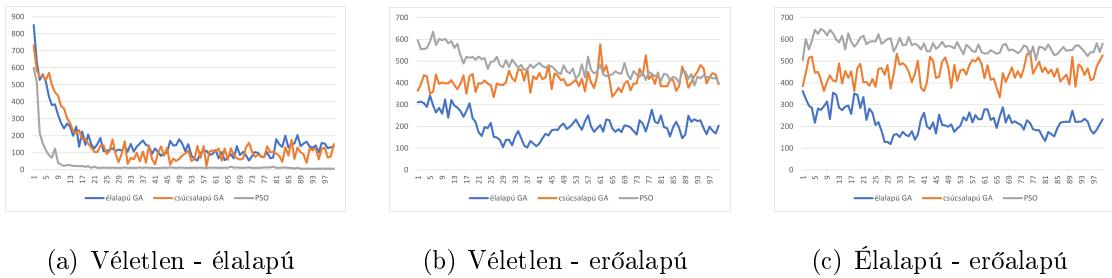
### Hibrid inicializálási módszerek

Továbbra is a  $H_{25,8,20}$  segítségével, de kizárálag véletlenszerű  $d$  értékek mellett vizsgálom, hogy kevert inicializálási megoldások jobban teljesítenek-e, mint a kizárolagosak. Ebben a kísérletben két különböző inicializációs megközelítés fele-fele arányban generálja az új megoldásokat.



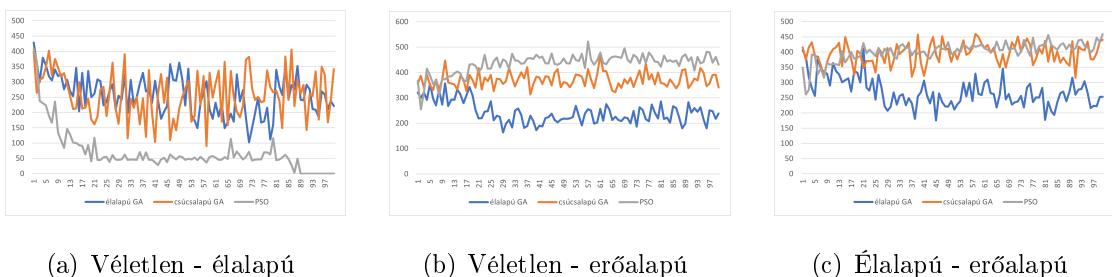
3.16. ábra. Inicializálás kevert módszerekkel - költségfüggvény

Összehasonlítva a tisztán egy módszert alkalmazó eredményekkel nem látszik, hogy a kevert módszerek különösebb előnyivel járnának. Kiemelendő tulajdonság azonban, hogy amennyiben az erőalapú módszert a véletlenszerűvel kombináljuk, akkor – az élalapú GA-hez hasonlóan – a PSO is képessé válik a kezdőértékeken túli konverenciára.



3.17. ábra. Inicializálás kevert módszerekkel - költségfüggvények átlaga

Érdemes megfigyelni, hogy a kevert véletlenszerű-élalapú inicializáció esetén érte el – a kísérleteim során először – a PSO a teljes konvergenciát a 200 iterációs lépése alatt. (Ez talán nem látszik annyira a költségfüggvényből – bár az is ellaposodik –, de a részecskék költségfüggvényeinek szórása már meggyőzhet minket róla.)

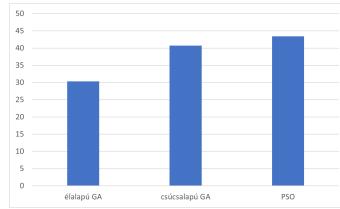


3.18. ábra. Inicializálás kevert módszerekkel - költségfüggvények szórása

### 3.2.5. Heurisztikák

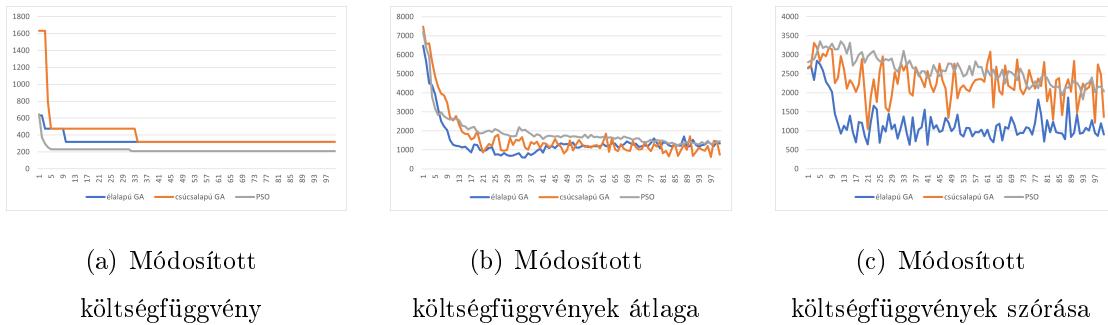
#### Tulajdonságok, mint mérőszámok

Említettem a heurisztikák kidolgozása során (3.1.4), hogy minél kisebb ugrások a költségfüggvényben javíthatják az algoritmusok hatásfokát. Ezt az állítást ellenőrzendő számos metrikát átfogalmaztam, hogy csak az optimális esetben értékelődjön ki nullára, minden más esetben egyre. Az ilyen – tulajdonságként kezelt metrikák – magukban foglalják a hamis pozitív tartalmazás, minimum távolság fenntartása, invalid szegmensmetszetek száma, legközelebbi szomszéd élszomszéd és fragmentálatlanság heurisztikákat. Következő kísérletként ezeknek a – kizárálag erre az egy esetre – átfogalmazott tulajdonságoknak a használatával vizsgálom, hogy ténylegesen romlik-e az algoritmusok teljesítménye. Az összehasonlíthatóság végett a kapott megoldásokat kiértékelem az eredeti mérőszámok szerint is.



3.19. ábra. Végeredmény az eredeti költségfüggvényben

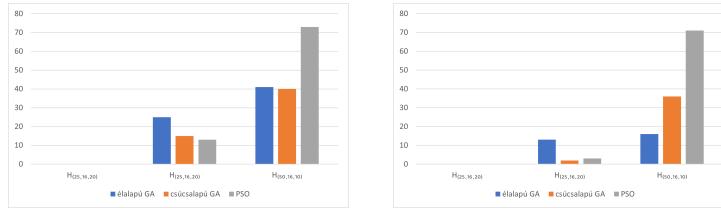
Megfelelően az előzetes feltevéseknek, megfigyelhető, hogy az összes algoritmus teljesítménye romlik. Konvergencia tekintetében egyedül a PSO megoldáspopulációja látszik javulni, azonban az is olyan lassan, hogy a konvergencia nem megfigyelhető az optimumértékben.



3.20. ábra. Tulajdonság alapú heurisztikák

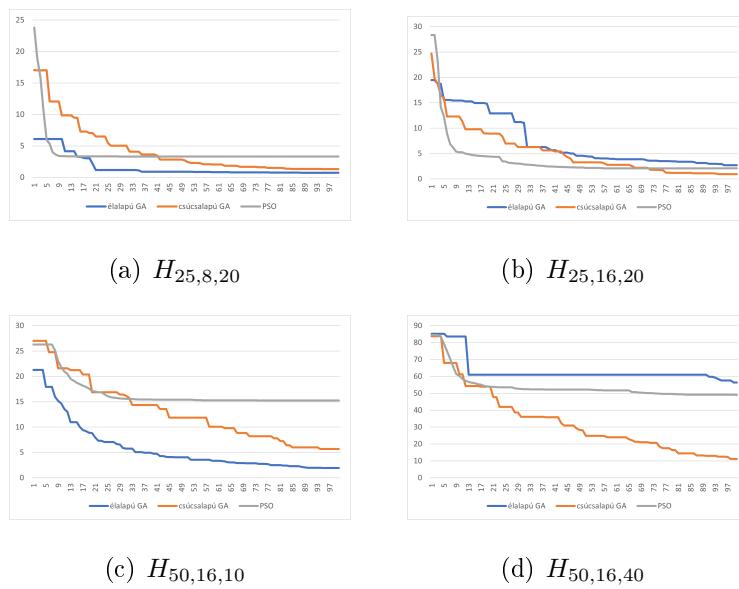
## Alternatív heurisztikák

Másik kérdésként merült fel, hogy a – rendkívül lassú – élmetszeteket leszámláló heurisztikák lecserélhetők-e egy gyorsabb alternatívára. Helyettesítő módszerként egy olyan helyettesítőt ajánlottam vizsgálni (legközelebbi szomszéd élszomszéd), amely a feltevésem szerint alkalmas lehet erre. Ebben a kísérletben a szegmensmetszetek számának és az invalid szegmensmetszetek számának súlyozását 0-ra állítottam, míg a legközelebbi szomszéd élszomszéd heurisztika súlyát 1000-re. Mivel az implementáció ezt nem teszi lehetővé, ezért itt nem közvetlenül a költségfüggvényeket hasonlítom össze, hanem a szegmensenkénti hamis pozitív csúcsok valós számát (nem a heuristikákát) vetem össze az eredeti súlyozás esetén előállóval. Még egyszer fontosnak tartom kiemelni, hogy ez a mérőszám legfeljebb  $(|V| - 1) * |E|$  értéket tud felvenni. A vizsgálatot a  $H_{25,8,20}$ ,  $H_{25,16,20}$  és  $H_{50,16,10}$  körében folytatom le.



3.21. ábra. Szegmensenkénti hibásan tartalmazott csúcsok száma

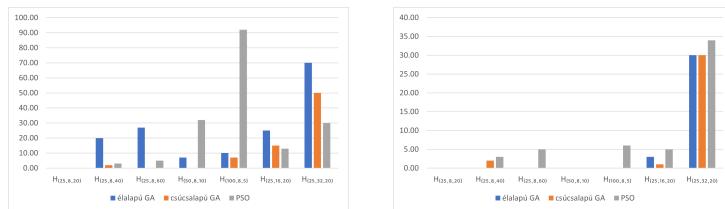
Meglepő módon nem csak a program futása lett nagyjából háromszor olyan gyors, de a vizsgált aspektusból még helyesebb megoldást is kaptunk. Az egymáshoz viszonyított költségfüggvények alapján a PSO valamivel rosszabbul teljesített, mint a másik két módszer, melyek egymáshoz hasonló eredményt adtak. Külön lefutattam az algoritmust egy nagyobb sűrűségű hipergráfra,  $H_{50,16,40}$ -re is, amely esetében azonban az élalapú GA teljesítménye nagyban elmaradt a csúcsalapúétól.



3.22. ábra. Költségfüggvény alternatív heurisztika esetén

### Heurisztikák teljesítménye

Meglehetősen nehéz értékelni, hogy egy adott ábrázolás mennyire tér el esztétikai kritériumrendszeruktól (például a well-formed követelményektől), azonban szemantikai szempontból ez a probléma nem áll fenn. Az alkalmazott heurisztikák természetesen csak akkor megfelelőek, ha alkalmasak a hipergráf helyes leképezésére.



3.23. ábra. Szemantikai hibák

### 3.2.6. Hipergráf konvolúciós hálózat

### 3.2.7. Futásidők

### 3.2.8. Összefoglalás

## 4. fejezet

# Konklúzió

# Irodalomjegyzék

- [1] C. Berge. *Hypergraphs*. North Holland Publishing Company, 1989. ISBN: 9780080880235.
- [2] Hooman Reisi Dehkordi és tsai. “Circular Graph Drawings with Large Crossing Angles”. *WALCOM: Algorithms and Computation*. Szerk. Subir Kumar Ghosh és Takeshi Tokuyama. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, 298–309. old. ISBN: 978-3-642-36065-7.
- [3] Janet M. Six és Ioannis G. Tollis. “Circular Drawings of Biconnected Graphs”. *Algorithm Engineering and Experimentation: International Workshop ALENEX’99 Baltimore, MD, USA, January 15–16, 1999 Selected Papers*. Szerk. Michael T. Goodrich és Catherine C. McGeoch. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, 57–73. old. ISBN: 978-3-540-48518-6. DOI: 10.1007/3-540-48518-X\_4. URL: [https://doi.org/10.1007/3-540-48518-X\\_4](https://doi.org/10.1007/3-540-48518-X_4).
- [4] Markus Eiglsperger, Sandor Fekete és Gunnar Klau. “Orthogonal Graph Drawing”. *Drawing Graphs: Methods and Models*, 121-171 (2001) 2025 (1999. jan.). DOI: 10.1007/3-540-44969-8\_6.
- [5] Brian Beckman. “Theory of spectral graph layout”. *vertex* 1 (1994), 2. old. URL: <http://research.microsoft.com/pubs/69611/tr-94-04.ps>.
- [6] Stephen Kobourov. “Force-Directed Algorithms”. 2013. jan., 383–408. old.
- [7] Walter Didimo, Giuseppe Liotta és Salvatore A. Romeo. “Topology-Driven Force-Directed Algorithms”. *Proceedings of the 18th International Conference on Graph Drawing*. GD’10. Konstanz, Germany: Springer-Verlag, 2010, 165–176. ISBN: 9783642184680.

- [8] Thomas M. J. Fruchterman és Edward M. Reingold. “Graph drawing by force-directed placement”. *Software: Practice and Experience* 21.11 (1991), 1129–1164. old. DOI: <https://doi.org/10.1002/spe.4380211102>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.4380211102>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380211102>.
- [9] Walter Didimo, Giuseppe Liotta és Salvatore A. Romeo. “Topology-Driven Force-Directed Algorithms”. *Proceedings of the 18th International Conference on Graph Drawing*. GD’10. Konstanz, Germany: Springer-Verlag, 2010, 165–176. ISBN: 9783642184680.
- [10] Bilal Alsallakh és tsai. “The State-of-the-Art of Set Visualization”. *Comput. Graph. Forum* 35.1 (2016. febr.), 234–260. ISSN: 0167-7055. DOI: 10.1111/cgf.12722. URL: <https://doi.org/10.1111/cgf.12722>.
- [11] Luana Micallef és Peter Rodgers. “eulerAPE: Drawing Area-Proportional 3-Venn Diagrams Using Ellipses”. *PloS one* 9 (2014. júl.), e101717. DOI: 10.1371/journal.pone.0101717.
- [12] Nathan Bailey és tsai. “Tissue-Specific Transcriptomics in the Field Cricket *Teleogryllus oceanicus*”. *G3 (Bethesda, Md.)* 3 (2013. febr.), 225–30. old. DOI: 10.1534/g3.112.004341.
- [13] P. Simonetto és D. Auber. “Visualise Undrawable Euler Diagrams”. *2008 12th International Conference Information Visualisation*. 2008, 594–599. old. DOI: 10.1109/IV.2008.78.
- [14] B. Alper és tsai. “Design Study of LineSets, a Novel Set Visualization Technique”. *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), 2259–2267. old. DOI: 10.1109/TVCG.2011.186.
- [15] Steven S. Skiena. *The Algorithm Design Manual*. 2nd. Springer Publishing Company, Incorporated, 2008. ISBN: 1848000693.
- [16] P. Chapman és tsai. “Visualizing Sets: An Empirical Comparison of Diagram Types”. *Diagrams*. 2014.

- [17] W. Freiler, K. Matkovic és H. Hauser. “Interactive Visual Analysis of Set-Typed Data”. *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), 1340–1347. old. DOI: 10.1109/TVCG.2008.144.
- [18] Margaret E. Baron. “A Note on the Historical Development of Logic Diagrams: Leibniz, Euler and Venn”. *The Mathematical Gazette* 53.384 (1969), 113–125. old. ISSN: 00255572. URL: <http://www.jstor.org/stable/3614533>.
- [19] Richard Brath. “Multi-Attribute Glyphs on Venn and Euler Diagrams to Represent Data and Aid Visual Decoding”. 2012. júl.
- [20] Anne Verroust és Marie-Luce Viaud. “Ensuring the Drawability of Extended Euler Diagrams for up to 8 Sets”. *Diagrammatic Representation and Inference*. Szerk. Alan F. Blackwell, Kim Marriott és Atsushi Shimojima. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, 128–141. old. ISBN: 978-3-540-25931-2.
- [21] Luana Micallef és Peter Rodgers. “eulerForce: Force-directed layout for Euler diagrams”. *Journal of Visual Languages & Computing* 25.6 (2014). Distributed Multimedia Systems DMS2014 Part I, 924 –934. old. ISSN: 1045-926X. DOI: <https://doi.org/10.1016/j.jvlc.2014.09.002>. URL: <http://www.sciencedirect.com/science/article/pii/S1045926X14000810>.
- [22] Peter Rodgers, Leishi Zhang és Helen Purchase. “Wellformedness Properties in Euler Diagrams: Which Should Be Used?”. *IEEE transactions on visualization and computer graphics* 18 (2012. júl.), 1089–100. old. DOI: 10.1109/TVCG.2011.143.
- [23] J. Flower, P. Rodgers és P. Mutton. “Layout metrics for Euler diagrams”. *Proceedings on Seventh International Conference on Information Visualization, 2003. IV 2003.* 2003, 272–280. old. DOI: 10.1109/IV.2003.1217990.
- [24] Mithileysh Sathiyaranarayanan és John Howse. “Well-matchedness in Euler Diagrams”. 2014. júl. DOI: 10.13140/2.1.2861.9524.
- [25] A. Blake és tsai. “Does the orientation of an Euler diagram affect user comprehension?”. *Proceedings: DMS 2012 - 18th International Conference on Distributed Multimedia Systems* (2012. jan.), 185–190. old.

- [26] Gem Stapleton és tsai. “Automatically drawing Euler diagrams with circles”. *Journal of Visual Languages & Computing* 23.3 (2012), 163 –193. old. ISSN: 1045-926X. DOI: <https://doi.org/10.1016/j.jvlc.2012.02.001>. URL: <http://www.sciencedirect.com/science/article/pii/S1045926X12000134>.
- [27] S. Chow. “Generating and Drawing Area-Proportional Euler and Venn Diagrams”. Dissz. University of Victoria, 2007. URL: <http://hdl.handle.net/1828/128>.
- [28] Stirling Chow és Frank Ruskey. “Drawing Area-Proportional Venn and Euler Diagrams”. *Graph Drawing*. Szerk. Giuseppe Liotta. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, 466–477. old. ISBN: 978-3-540-24595-7.
- [29] Stirling Chow és Frank Ruskey. “Towards a General Solution to Drawing Area-Proportional Euler Diagrams”. *Electronic Notes in Theoretical Computer Science* 134 (2005). Proceedings of the First International Workshop on Euler Diagrams (Euler 2004), 3 –18. old. ISSN: 1571-0661. DOI: <https://doi.org/10.1016/j.entcs.2005.02.017>. URL: <http://www.sciencedirect.com/science/article/pii/S1571066105050395>.
- [30] Paolo Simonetto, David Auber és Daniel Archambault. “Fully Automatic Visualisation of Overlapping Sets”. *Computer Graphics Forum* 28.3 (2009), 967–974. old. DOI: <https://doi.org/10.1111/j.1467-8659.2009.01452.x>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2009.01452.x>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01452.x>.
- [31] Andrew Blake és tsai. “How Should We Use Colour in Euler Diagrams?”. *Proceedings of the 7th International Symposium on Visual Information Communication and Interaction*. VINCI ’14. Sydney NSW, Australia: Association for Computing Machinery, 2014, 149–158. ISBN: 9781450327657. DOI: [10.1145/2636240.2636838](https://doi.org/10.1145/2636240.2636838). URL: <https://doi.org/10.1145/2636240.2636838>.
- [32] Andrew Fish, Babak Khazaie és Chris Roast. “User-comprehension of Euler diagrams”. *Journal of Visual Languages & Computing* 22.5 (2011), 340 –

354. old. ISSN: 1045-926X. DOI: <https://doi.org/10.1016/j.jvlc.2011.01.002>. URL: <http://www.sciencedirect.com/science/article/pii/S1045926X11000036>.
- [33] Andrew Blake és tsai. “The Impact of Shape on the Perception of Euler Diagrams”. *Diagrammatic Representation and Inference*. Szerk. Tim Dwyer, Helen Purchase és Aidan Delaney. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, 123–137. old. ISBN: 978-3-662-44043-8.
- [34] G. Stapleton és tsai. “Inductively Generating Euler Diagrams”. *IEEE Transactions on Visualization and Computer Graphics* 17.1 (2011), 88–100. old. DOI: 10.1109/TVCG.2010.28.
- [35] M. Wang és tsai. “SketchSet: Creating Euler diagrams using pen or mouse”. *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2011, 75–82. old. DOI: 10.1109/VLHCC.2011.6070382.
- [36] Roger Fletcher. “On the Barzilai-Borwein Method”. *Optimization and Control with Applications*. Szerk. Liqun Qi, Koklay Teo és Xiaoqi Yang. Boston, MA: Springer US, 2005, 235–256. old. ISBN: 978-0-387-24255-2.
- [37] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. ISBN: 0262082136.
- [38] Warren Hare, Julie Nutini és Solomon Tesfamariam. “A survey of non-gradient optimization methods in structural engineering”. *Advances in Engineering Software* 59 (2013), 19 –28. old. ISSN: 0965-9978. DOI: <https://doi.org/10.1016/j.advengsoft.2013.03.001>. URL: <http://www.sciencedirect.com/science/article/pii/S0965997813000288>.
- [39] J. Kennedy és R. Eberhart. “Particle swarm optimization”. *Proceedings of ICNN'95 - International Conference on Neural Networks*. 4. köt. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.
- [40] Yuhui Shi és B.Gireesha Obaiahnahatti. “A Modified Particle Swarm Optimizer”. 6. köt. 1998. jún., 69 –73. old. ISBN: 0-7803-4869-9. DOI: 10.1109/ICEC.1998.699146.

- [41] M. Zambrano-Bigiarini, M. Clerc és R. Rojas. “Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements”. *2013 IEEE Congress on Evolutionary Computation*. 2013, 2337–2344. old. DOI: 10.1109/CEC.2013.6557848.
- [42] F. Ahmad és tsai. “Performance comparison of gradient descent and Genetic Algorithm based Artificial Neural Networks training”. *2010 10th International Conference on Intelligent Systems Design and Applications*. 2010, 604–609. old. DOI: 10.1109/ISDA.2010.5687199.
- [43] Edmund Ronald és Marc Schoenauer. “Genetic Lander: An Experiment in Accurate Neuro-Genetic Control”. *Proc. 3rd Conf. Parallel Problem Solving from Nature*. Springer-Verlag, 1994, 452–461. old.
- [44] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
- [45] M. Gori, G. Monfardini és F. Scarselli. “A new model for learning in graph domains”. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005*. 2 (2005), 729–734 vol. 2.
- [46] David K Duvenaud és tsai. “Convolutional Networks on Graphs for Learning Molecular Fingerprints”. *Advances in Neural Information Processing Systems 28*. Szerk. C. Cortes és tsai. Curran Associates, Inc., 2015, 2224–2232. old. URL: <http://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints.pdf>.
- [47] Thomas Kipf és M. Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. *ArXiv* abs/1609.02907 (2017).
- [48] Michaël Defferrard, Xavier Bresson és Pierre Vandergheynst. “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”. *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, 3844–3852. ISBN: 9781510838819.
- [49] Wen-Ch’ing Winnie Li és Patrick Solé. “Spectra of Regular Graphs and Hypergraphs and Orthogonal Polynomials”. *European Journal of Combinatorics* 17.5 (1996), 461–477. old. ISSN: 0195-6698. DOI: 10.1006/eujc.1996.0040.

- [50] Andries E. Brouwer és Willem H. Haemers. *Spectra of Graphs*. New York, NY, 2012. DOI: 10.1007/978-1-4614-1939-6.
- [51] Anand Louis. “Hypergraph Markov Operators, Eigenvalues and Approximation Algorithms”. *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*. STOC ’15. Portland, Oregon, USA: Association for Computing Machinery, 2015, 713–722. ISBN: 9781450335362. DOI: 10.1145/2746539.2746555. URL: <https://doi.org/10.1145/2746539.2746555>.
- [52] T.-H. Hubert Chan és tsai. “Spectral Properties of Hypergraph Laplacian and Approximation Algorithms”. *J. ACM* 65.3 (2018. márc.). ISSN: 0004-5411. DOI: 10.1145/3178123. URL: <https://doi.org/10.1145/3178123>.
- [53] Yifan Feng és tsai. “Hypergraph Neural Networks”. *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), 3558–3565. old. DOI: 10.1609/aaai.v33i01.33013558. URL: <https://ojs.aaai.org/index.php/AAAI/article/view/4235>.
- [54] Naganand Yadati és tsai. “HyperGCN: Hypergraph Convolutional Networks for Semi-Supervised Classification”. (2018. szept.).
- [55] Sambaran Bandyopadhyay, Kishalay Das és M. Murty. “Line Hypergraph Convolution Network: Applying Graph Convolution for Hypergraphs”. (2020. febr.).
- [56] Loc Tran. “Directed Hypergraph Neural Network”. *Journal of Advanced Research in Dynamical and Control Systems* 12 (2020. márc.), 1434–1441. old. DOI: 10.5373/JARDCS/V12SP4/20201622.
- [57] Twan Laarhoven. “L2 Regularization versus Batch and Weight Normalization”. (2017. jún.).
- [58] Kai Hormann és Alexander Agathos. “The point in polygon problem for arbitrary polygons”. *Computational Geometry* 20.3 (2001), 131 –144. old. ISSN: 0925-7721. DOI: [https://doi.org/10.1016/S0925-7721\(01\)00012-8](https://doi.org/10.1016/S0925-7721(01)00012-8). URL: <http://www.sciencedirect.com/science/article/pii/S0925772101000128>.