

TDK dolgozat

Papp György

DEBRECEN

2014

DEBRECENI EGYETEM
INFORMATIKAI KAR
KOMPUTERGRAFIKA ÉS KÉPFELDOLGOZÁS TANSZÉK

Nagyméretű mátrixok vizualizációja a weben hiperbolikus geometriai alapokon

Témavezető:

Kunkli Roland
tanársegéd

Készítette:

Papp György
programtervező informatikus MSc
1. évfolyam

DEBRECEN

2014

Tartalomjegyzék

1. Bevezetés	4
1.1. Az adatvizualizáció népszerűsége és fontossága	4
1.2. Adatvizualizációs technikák	5
1.2.1. Kör alapú megoldások	6
1.2.2. Circular layout	7
2. Circos	9
2.1. A program népszerűségének okai	10
2.2. A Circos alkalmazás hiányosságai	10
2.2.1. Célkitűzések	14
3. Önálló eredményeink, megoldásaink a Circos problémáira	15
3.1. A Poincaré-féle körmodell ismertetése	15
3.1.1. Apollóniusz-kör	17
3.2. A matematikai modell használatából fakadó előnyök	19
3.3. Kapcsolatok rendezése	20
3.4. Kapcsolatok felosztása	21
3.4.1. A felosztás menete	21
4. Az alkalmazásunk	24
4.1. Irány a web	25
4.2. Technológiai újdonságok	25
4.3. A látható előny	27
5. A saját alkalmazásunk és a Circos összehasonlítása	28
5.1. Hasonlóságok	28
5.2. Különbségek	29
5.3. Összehasonlítás képekben	29
6. Összefoglalás	34

A. A vizualizáció alapját képező táblázatok	40
B. További összehasonlító ábrák	43

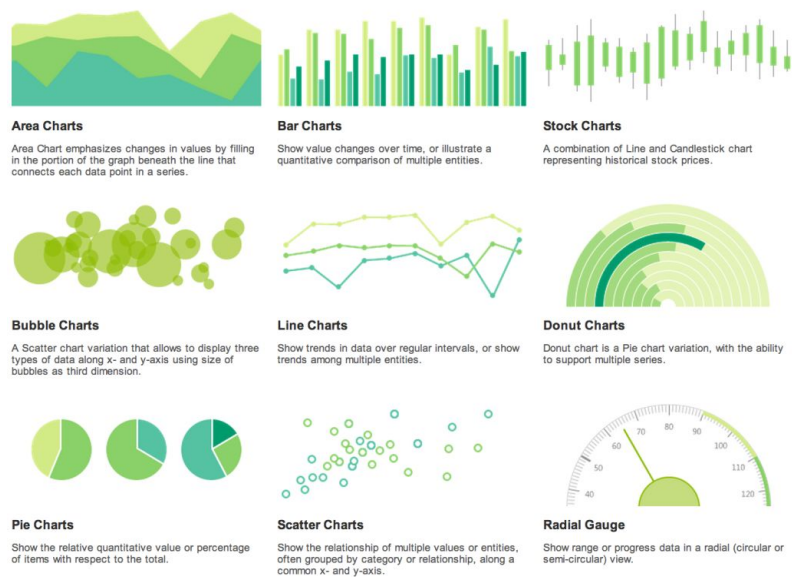
1. fejezet

Bevezetés

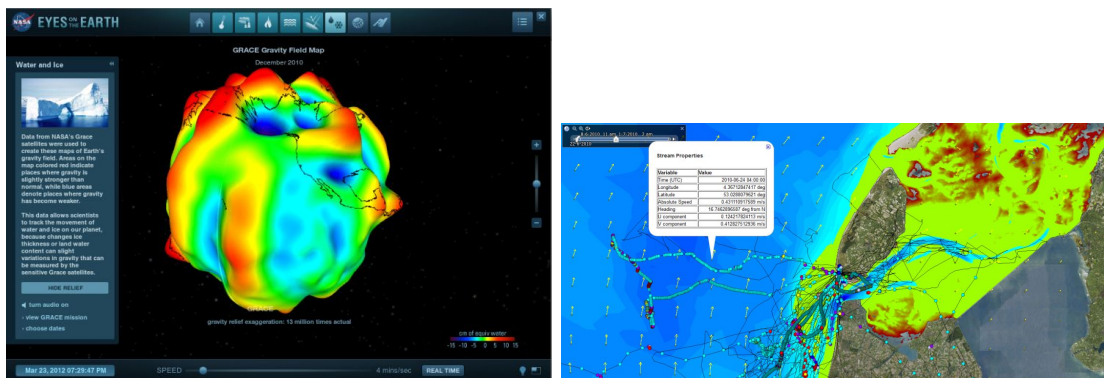
1.1. Az adatvizualizáció népszerűsége és fontossága

Az informatikában és a szorosan hozzá kötődő területeken bekövetkezett fejlődésnek köszönhetően lehetőségünk nyílt arra, hogy egyre több adatot gyűjtsünk, kezeljünk és tároljunk. Azonban az, hogy az adatokat hogyan dolgozzuk fel úgy, hogy a kapott végeredmény hasznos legyen a számunkra, már nem egyértelmű. Egy látványos és remek eszköz erre a feladatra az adatvizualizáció, amivel a rendelkezésünkre álló adatokból úgy tudunk vizualizációt készíteni, hogy az elkészült megjelenítésből számunkra hasznos információkhoz tudunk hozzájutni.

Az adatvizualizációt széles körben használják, kezdve az általános iskolásoktól a kutatókon keresztül a gazdasági elemzőikig, hogy a rendelkezésre álló adatokat olyan formában reprezentálják, hogy a megjelenítés informatív legyen. Általánosságban a lakosság az egyszerű diagramokat (oszlop-, kör-, vonaldiagram) (lásd 1.1. ábra) használja, hogy adatvizualizációt készítsen. Ezek a diagramok azért lehetnek ilyen népszerűek, mivel nagyon sok módon felhasználhatók azáltal, hogy változatos adatokat lehet velük megjeleníteni, valamint jól testreszabhatók. A tudomány területén dolgozó kutatók azonban már nem egy általánosan alkalmazott megjelenítést használnak, mint például az előbb említett diagramok, hanem egy olyat ami a legjobban megfelel a saját igényeiknek (lásd 1.2. ábra). Ez azzal magyarázható, hogy nagyon különbözőek az adatok, amelyeket vizualizálni kell, illetve a vizualizációval másfajta összefüggéseket, információkat akarunk megtudni az adatokból.



1.1. ábra. Általánosságban használt diagramok [22]

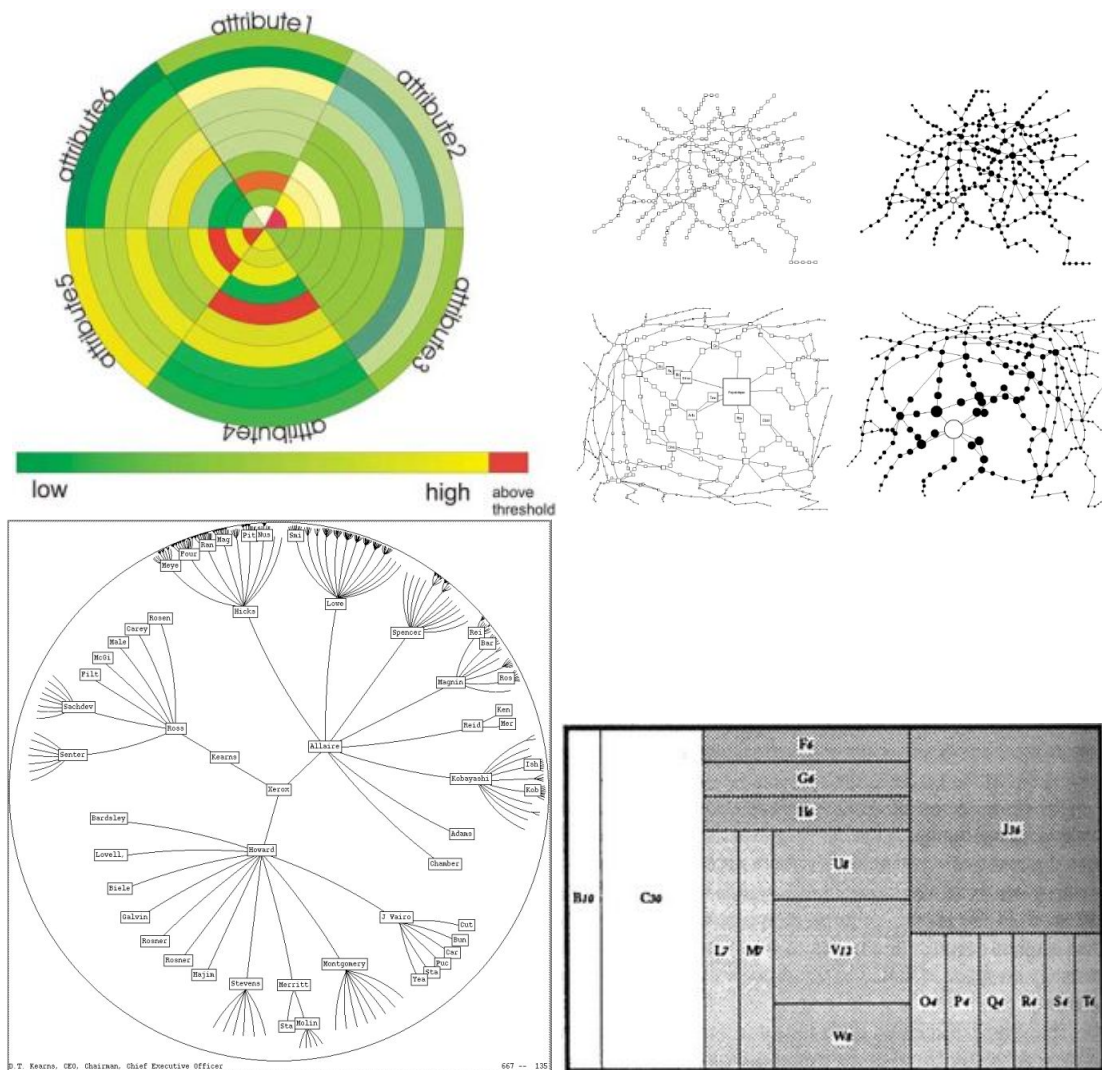


1.2. ábra. NASA Eyes on the Earth projekt [20] (bal oldali kép), SURFsara FlySafe projekt [19] (jobb oldali kép)

1.2. Adatvizualizációs technikák

Láthatjuk, hogy nem létezik egységes, univerzális megjelenítési technika, amelynek a segítségével egy varázsütésre tökéletes vizualizációt kaphatnánk. Ehelyett technikák egész tárháza áll rendelkezésünkre, amiből kiválaszthatjuk a nekünk legmegfelelőbbet, hogy az elvárásainknak megfelelő ábrázolást tudjunk készíteni. A teljesség igénye nélkül röviden ismertetünk párat.

A *TreeMap* [6] (lásd 1.3. ábra) egy térkitöltő technika, amelynek a segítségével hierarchikusan strukturált információkat tudunk megjeleníteni. Az egyes információknak egy téglalapot feleltetünk meg, a mérete pedig a hierarchiában alatta lévő elemek méretétől



1.3. ábra. A *CircleView* [9] (bal felső kép), *FishEye* [18] (jobb felső kép), *Hyperbolic Trees* [11] (bal alsó kép) és a *Cone trees* [3] (jobb alsó kép) technikák.

függ. A *FishEye* [18] (lásd 1.3. ábra) technika alapját a halszemoptika képezi, ami egy nagyon széles látószögű objektív, a közeli tárgyakat felnagyítja, a távoliakat pedig leicsinyíti. Ezzel a vizualizációban azt lehet elérni, hogy a lokális részletek kiemelésével nem veszítjük el a globális kontextust, hanem csak háttérbe kerülnek.

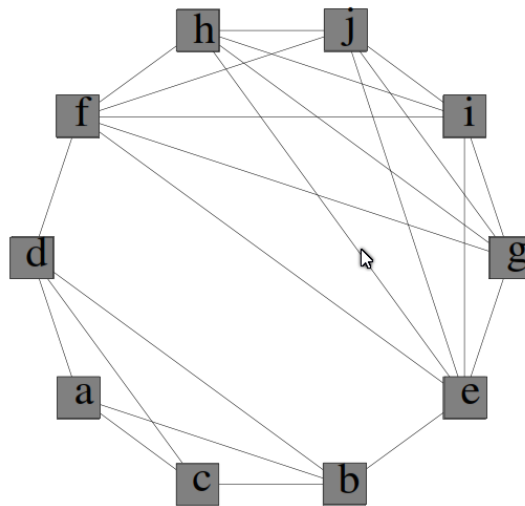
1.2.1. Kör alapú megoldások

A következő technikák mindegyike alapelemként kezeli a kört. A *Radial Hierarchical Visualization* [8] és *CircleView* [9] (lásd 1.3. ábra) vizualizációs technikák nem térnek el sokban egymástól. Mindkét módszernél egy kör szolgál alapul az információk megjelenítéséhez. Ezt a kört osztjuk fel több részre, ami kétféle módon történhet. Először úgy, hogy

a körön belül sávokat alakítunk ki, amelyeknek különböző értékeket tudunk megfeleltetni, másodszor pedig a kört körcikkre osztjuk és a körcikknek feleltetünk meg valamilyen adatot. A *Hyperbolic Trees* [11] (lásd 1.3. ábra) hasonló az előző két technikához, mivel itt is a vizualizáció alapját egy kör képezi, olyan módon, hogy ezen a körön belül egy gráffal jelenítjük meg az adatok közötti kapcsolatokat a hiperbolikus sík tulajdonságait és előnyeit kihasználva. A *FishEye*-t és *CircleView*-t kivéve az előbb említett technikák használatával nagy mennyiségű hierarchikus információ ábrázolására van lehetőségünk.

1.2.2. Circular layout

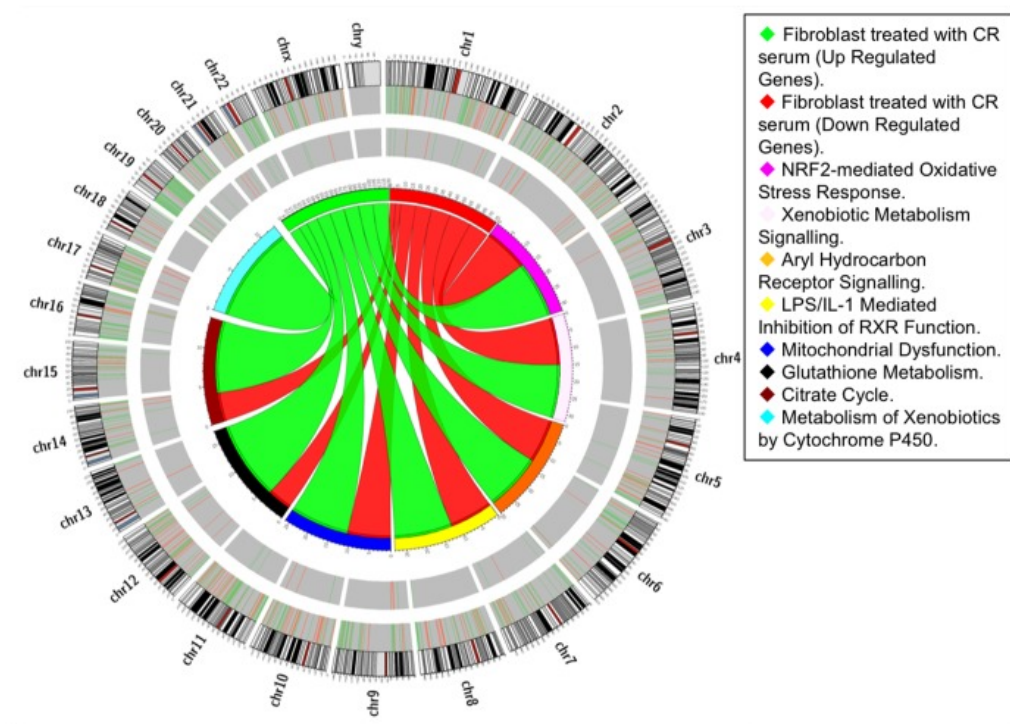
A *Circular layout* [2] (lásd 1.4. ábra) módszerrel az adatokat egy kör mentén tudjuk elhelyezni, és köztük a körön belül vonalakkal tudjuk reprezentálni a kapcsolatokat. Előnye, hogy a szemünknek kevesebbet kell mozognia az egyes adatok között, mint lineáris ábrázolás esetén. Kevésbé érzékeny a lokális mintákra, gyengíti az adatok sorrendjéből fakadó félreértéseket, valamint semleges a forgatásra nézve. Azt már tudjuk, hogy a körön alapuló vizualizációval nagy mennyiségű hierarchikus információt tudunk megjeleníteni, most pedig azt láthattuk, hogy szintén körön alapuló vizualizációval lehet nem hierarchikus nagy mennyiségű adatot is vizualizálni. A *Circos* [10] (lásd 1.5. ábra) a körvonal mentén való



1.4. ábra. *Circular layout* [21]

megjelenítést (*Circular layout*) használja arra, hogy nagy mennyiségű adatokat jelenítsen meg. Egy népszerű szoftverről [1] [17] [16] van szó, amit mind a tudományos életben, mind a hétköznapiakban sokan használnak. Azonban a programnak a vizualizáció elkészítésében hiányosságai, problémái vannak. A dolgozat elkészítésének célja az volt, hogy megoldást

találjunk a Circosban felmerülő problémákra, illetve a megtalált megoldásokra alapozva egy szoftvert készítsünk, amellyel reprezentálni tudjuk az elért eredményeket.



1.5. ábra. Circossal készült vizualizáció [17]

2. fejezet

Circos

A Circosnál már megfigyelhető, hogy nemcsak a genomi adatok megjelenítésén van a hangsúly, hanem azon is, hogy univerzálisan bármilyen adatot vizualizálni tudjunk az alkalmazással. A szoftver *Perl* scriptnyelven íródott és GPL licenc alatt érhető el. Ez azt jelenti, hogy szabadon felhasználható, módosítható. A program bármilyen platformon futtatható, amelyen megtalálható a *Perl* futtatókörnyezetet, illetve a szoftverhez szükséges modulok, de ha ezek nem állnak rendelkezésre alapértelmezetten, akkor saját magunk fel tudjuk telepíteni őket, mivel minden platformra elérhetők. Ezek után parancssorból az előre megadott mintaadatokból már vizualizációt is készíthetünk. Az alkalmazást konfigurációs fájlokon keresztül tudjuk vezérelni, azonban ehhez meg kell tanulnunk a programhoz tartozó API-t. Az alkalmazás használható az API bővebb ismerete nélkül is, mivel kapunk egy alap konfigurációs fájlt, azonban ha minden részletre kiterjedően szeretnénk a program kimenetét befolyásolni, akkor annak mélyebb ismeretére van szükségünk. A beállítások sokszínűségére nem lehetünk panasszal, mivel a diagram minden területét tudjuk befolyásolni, emellett lehetőségünk van dinamikus szabályok megadására, amik futás közben az adatok alapján értékelődnek majd ki és kerülnek alkalmazásra. Az információk közötti összefüggések megjelenítésére ún. kapcsolatokat használ az alkalmazás. Két Bézier-görbe által közbezárt terület reprezentál egy kapcsolatot (2.1. ábra). A program kimenete egy SVG fájl, valamint egy 2000×2000 -es felbontással rendelkező kép.

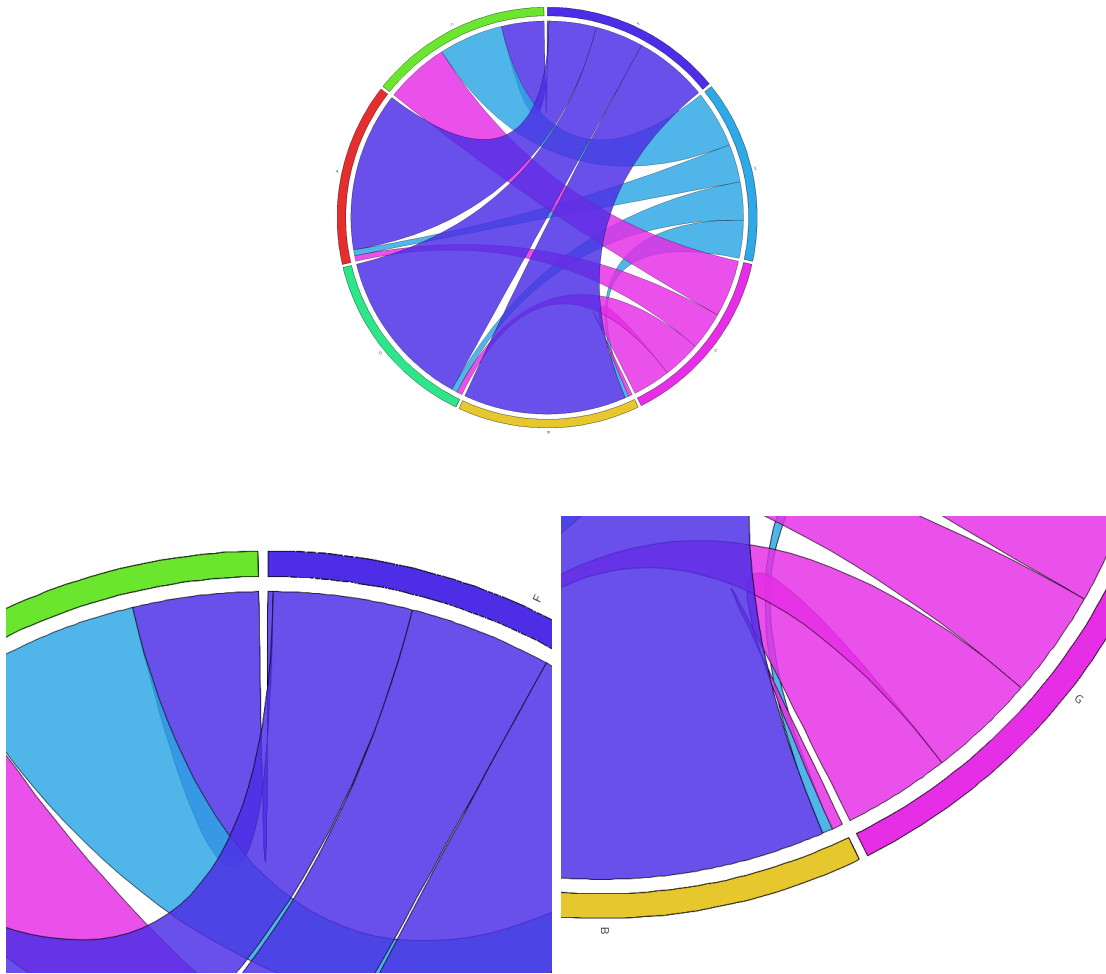
A szoftverhez több eszközt is nyújtanak. Egy ilyen a *table viewer* is, amivel táblázatos adatokat tudunk generálni, illetve azokból olyan konfigurációs fájlokat készíteni, amit a Circossal később meg tudunk jeleníteni. Ehhez külön a table viewert is tudjuk konfigurálni. Az eszközhöz készült egy webalkalmazás is, a Circos Table Viewer [14], aminek a segítségével kevesebb beállítási lehetőség mellett tudunk táblázatokat vizualizálni.

2.1. A program népszerűségének okai

Nem lehet egyszerűen megmagyarázni azt, hogy miért terjedt el a program a kutatók körében, így csak az általunk lehetségesnek tartott okokat tudjuk felvázolni. Azzal, hogy az alkalmazást Perl nyelven írták meg, ezzel biztosítva a multiplatform támogatást, lehetővé tették, hogy széles körben használható legyen. Az előre tartalmazott adat- és konfigurációs fájlok jó kiindulási alappal szolgálnak a szoftverrel valós ismerkedéshez, egyéni konfigurációk elkészítéséhez. A Circos honlapján [13] egy telepítési segédanyag és programdokumentáció áll a rendelkezésünkre, ezek a kezdetekben nagyon hasznosnak tudnak bizonyulni. A részletekbe menő testreszabási lehetőségek is fontos szempontot jelentenek, amiket nem hagyhatunk ki a felsorolásból, hiszen így még szebb ábrákat állíthatunk elő. Nem utolsó szempont, hogy mivel a szoftvert GPL licenc alatt érhetjük el, emiatt az bárki számára ingyenesen letölthető és szabadon módosítható. Mivel a circossal megjeleníthető adatokra nincs korlátozás, ez nagy előnyt jelent a versenytársakkal szemben. Utoljára pedig a Circos mögött álló közösség az, ami még inkább hozzájárul magának a szoftvernek az elterjedéséhez, mivel így nagyobb esély van arra, hogy a problémát, amivel találkozunk a konfiguráció elkészítése alatt, valaki már megoldotta, illetve, hogy a fejlesztők nem fogják magára hagyni a projektet és vele együtt a hatalmas felhasználói táborot.

2.2. A Circos alkalmazás hiányosságai

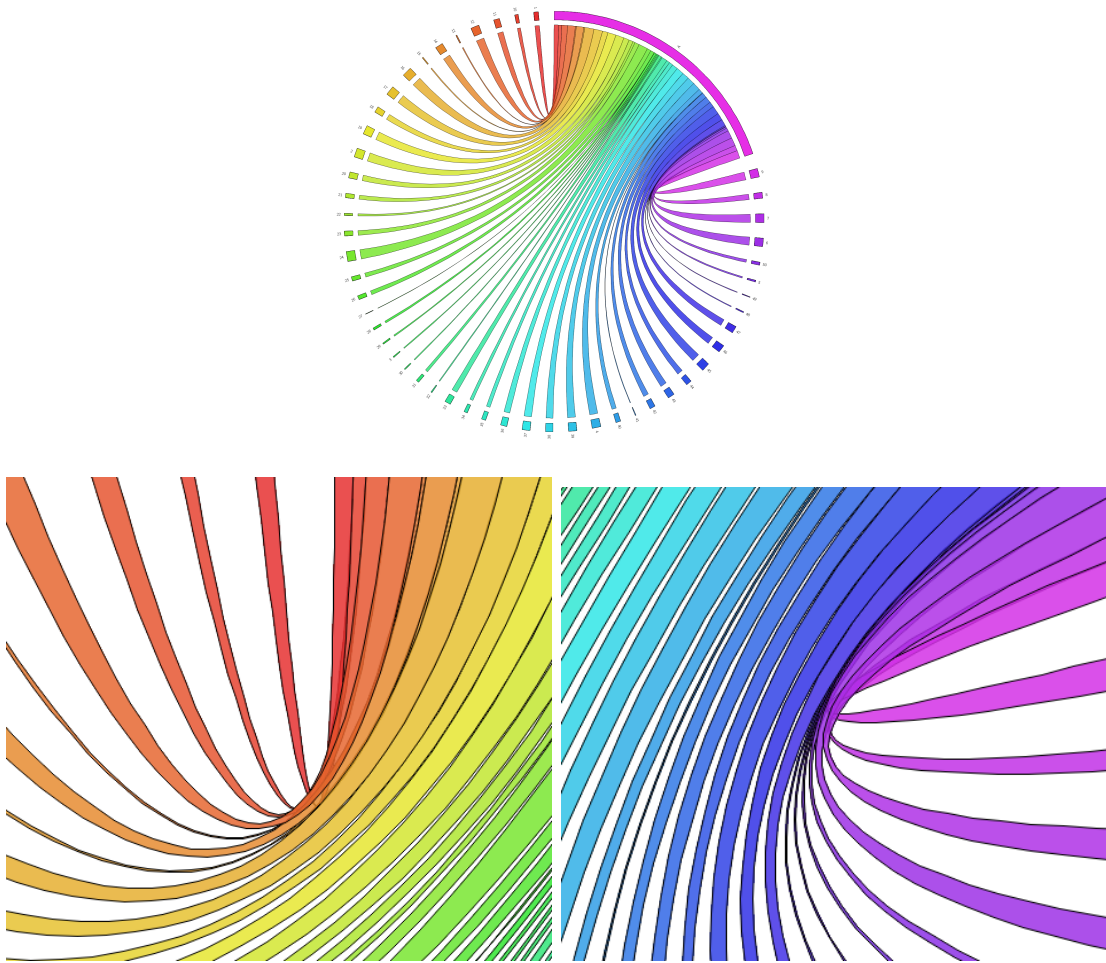
Azonban az előnyök mellett ott rejlenek a különböző hiányosságok, illetve problémák is. A kapcsolatok fontosak az összefüggések megértésében, azonban a megjelenítésük nem mindig tökéletes, emiatt sérül a diagram átláthatósága és az információ, amit közölni szeretnénk. Egy ilyen probléma merül fel, amikor két, a körvonalon egymáshoz közel elhelyezkedő, jelentősen különböző körívhosszal rendelkező körgyűrűcikk között szeretnénk kapcsolatot megjeleníteni. Ebben az esetben a kapcsolatot reprezentáló két Bézier-görbe metszi egymást (lásd 2.1. ábra). Ennek a végeredménye egy eltorzult kapcsolat lesz, ami a fent említettekhez vezet. Sajnos a görbeparaméterek módosításával ebben az esetben nem lehet olyan állapotot produkálni, ahol a kapcsolatot reprezentáló két görbe nem metszi egymást. Megpróbáltunk egy olyan beállítást találni, ahol a legkevésbé jelentkezik ez a hiba, azonban ebben az esetben a diagram többi részét rontottuk el a paraméterek ilyen mértékű radikális módosításával. Kipróbáltuk még azt is, hogy mi lenne, ha nem az egész diagramra alkalmaznánk az egyes paraméter beállításokat, hanem csak bizonyos linkekre. A dinamikus szabályok segítségével részben be tudjuk határolni a problémás kapcsolatokat, azonban mivel csak közelítőleg tudunk megoldást mondani a metszés problémájára,



2.1. ábra. A kapcsolatok „önmetszésének” problémája (fent) a Circosban, a problémás részek kiemelésével (lent).

így csak részlegesen tudnánk azon javítani. A paraméterek próbálgatása miatt ez a javítás időigényes, és sokszor a kapott eredmény nagyon messze van attól, amit vártunk.

Egy másik probléma is adódik a kapcsolatokkal, ha azok közel helyezkednek el és egymás mellett haladnak. Ebben az esetben a közel elhelyezkedő linkek egymásba vághatnak, vagy eltakarhatják egymást, illetve annyira összezsúfolódhatnak, hogy elveszítik a jelentésüket (lásd 2.2. ábra). Emiatt nem lehet meghatározni, hogy mely elemek között van kapcsolat, csupán a linkek alakjának az alapján. A paraméterek módosításával itt sem tudunk megfelelően jó eredményt elérni, mivel nincsen olyan egységes megoldás, ami a teljes diagramra helyes megjelenítést adna. Emellett a feltételes módosítások nem nyújtanak jó eredményt, és a paraméterbeállításokkal, feltételekkel történő kísérletezés időigényes.



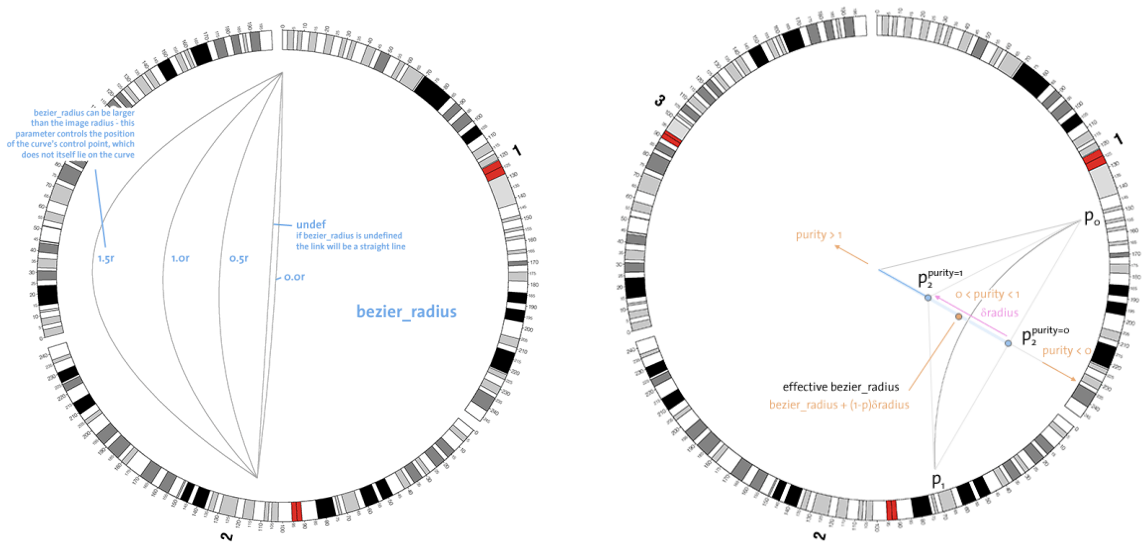
2.2. ábra. A kapcsolatok összezsúfolódásának problémája (fent) a Circosban, a problémás részek kiemelésével (lent).

A három paraméter, amelyeknek a módosításával megpróbáltuk elérni, hogy az alkalmazás a fenti helyzetekben jobb eredményt adjon:

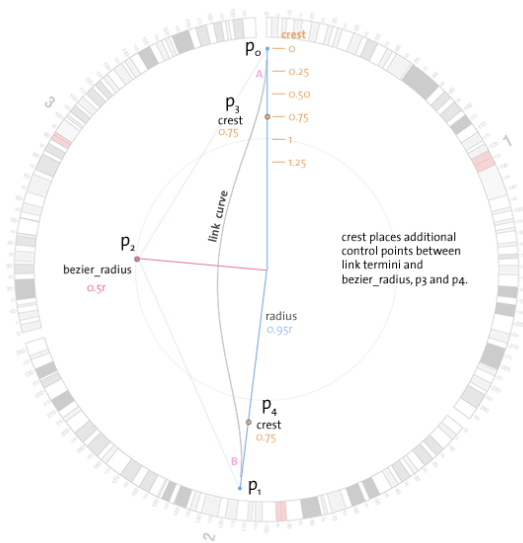
- *bezier_radius*
- *bezier_radius_purity*
- *crest*

A 2.3. és a 2.4. ábrákon látható, hogy a paraméterek értékének módosításával hogyan lehet változtatni a görbéket [12].

Azokban az esetekben, amelyiknél nem fontos az egyes környűricikkeknek és az azokon belül elhelyezkedő linkeknek a sorrendje, rendezhetnénk a kapcsolatokat úgy, hogy a lehető legkevesebb metszés legyen köztük a diagramban. Ehhez jó kiindulási pont lenne, ha



2.3. ábra. A *bezier_radius* és a *bezier_radius_purity* paraméter hatása a görbére [12]



2.4. ábra. A *bezier_radius* és a *crest* paraméter hatása a görbére [12]

meg tudnánk határozni, hogy két kapcsolat mikor metszi egymást. Ez a Circos esetében körülményes, mivel a kapcsolatokat leíró Bézier-görbék metszéspontjait nehéz meghatározni. Tehát ebből az irányból nem tudnánk megközelíteni a problémát, így megfelelő megoldást sem nyújthatnánk rá.

Hiányosságként emelném ki, hogy az ábra finomhangolása nehézkes és időigényes feladat, mivel minden egyes konfigurációs fájlban történt változtatás esetén újra el kell végezni a

kép generálását. A parancssori futtatásnak és a konfigurációs fájlkon keresztüli beállítások megadásának a fent említett előnyei mellett megvannak a hátrányai is. Ez a hátrány leginkább a kevés informatikai tudással rendelkező felhasználók körében jelentkezik, akiknek nehéz megtanulni, megszokni a program kezelését. Tehát az alkalmazás hátránnyal indul a grafikus felhasználói felülettel ellátott rendszerekkel szemben, az informatikában kevésbé jártas felhasználók körében. Emellett magának a programnak a telepítése sem megy ugyanolyan zökkenőmentesen minden rendszeren. GNU/Linux alapú rendszereken lehet a legkevesebb problémába ütközni, viszont Windows alapú rendszereken olyan hibák jöhetnek elő, amelyekre nincsen megoldás feltüntetve a dokumentációban, így egyéb forrásból kell azt megtalálnunk. Felhasználói interakció csak az automatikusan kiértékelődő szabályokban érhető tetten, egyébként nincs lehetőségünk a kimenet változtatására futás közben. Utolsó sorban pedig, ha a genomi adatokat figyelmen kívül hagyjuk és csak az általános adatok megjelenítését tekintjük, akkor szomorúan azt tapasztalhatjuk, hogy csak egész számokkal képes a szoftver dolgozni.

2.2.1. Célkitűzések

A Circosnak sok előnye van és ezek mind hozzájárultak ahhoz, hogy egy népszerű alkalmazás legyen belőle, azonban mint a legtöbb program, ez sem hibátlan. Röviden összefoglalva a hibákat:

- A kapcsolatok megjelenítésében vannak problémás esetek, amikor nehezen átlátható ábrákat kapunk.
- Felhasználói interakciók hiánya.
- Kevés informatikai tudással nehézkes a használat.

Munkánk során ezeket a hibákat szeretnénk volna orvosolni. Mindezt úgy akartuk megvalósítani, hogy a Circosnak csak azokat a funkcionalitásait tartjuk meg, amik fontosak a problémák szempontjából és a többit pedig elhagyjuk, hogy egy olyan funkcióhalmazt kapjunk, amiben a problémák még fennállnak, és a jövőben jó összehasonlítási alapot biztosítanak majd a számunkra. Mivel a Circosnak már volt egy funkcióiban redukált alkalmazása, a Circos Table Viewer, így a megoldás tálcán kínálkozott, mégpedig, hogy a továbbiakban ezt az alkalmazást vegyük összehasonlítási alapként.

3. fejezet

Önálló eredményeink, megoldásaink a Circos problémáira

A már felsorolt problémákra olyan megoldást szerettünk volna adni, amely a gyakorlatban is megállja a helyét és reprezentálja, hogy ténylegesen orvosoltuk a hibákat. Ezekhez egy alkalmazás elkészítésében láttuk a megoldást.

Mivel azt láttuk, hogy a görbe paraméterezésének segítségével nem tudjuk megoldani a Bézier-görbékkel megadott kapcsolatoknál felmerülő problémákat, ezért egy új modellre volt szükségünk. Olyat szerettünk volna keresni, melynek alapelemei a Bézier-görbéknél könnyebben kezelhető görbeívek. További előnyöket is várva jutottunk el a hiperbolikus sík Poincaré-féle körmodelljéig.

3.1. A Poincaré-féle körmodell ismertetése

Ebben a fejezetben a szakdolgozatomban [5] már ismertetett matematikai modellek leírása [25] található kisebb változtatással.

Ha a hiperbolikus geometria axiómarendszerére modellt szeretnénk adni, interpretálnunk kell az abban foglalt alapfogalmakat és meg kell mutatnunk, hogy az axiómák a modellként szánt matematikai rendszerünkben tételként jelennek meg.

Tekintsük alapul szolgáló rendszernek $\mathbb{R}^2$ geometriáját, s nevezzük E-köröknek $\mathbb{R}^2$ egydimenziós lineáris sokaságait és köreit. Legyen $O = (0,0)$ és tekintsük az alábbi két halmazt:

$$\omega = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 = 1\}, \quad \text{int}\omega = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 < 1\}.$$

Ezen jelölések bevezetése után definiáljuk, mit nevezünk pontnak és egyenesnek a modellben.

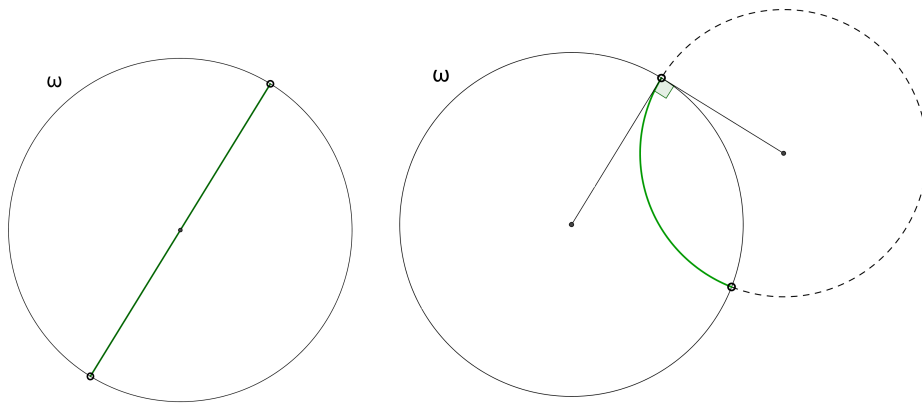
Definíció: $int\omega$ elemeit pontoknak nevezzük, egyenes alatt pedig egy ω -t merőlegesen metsző E-kör és $int\omega$ metszetét értjük.

Belátható, hogy a modellben teljesülnek az illeszkedési axiómák és a hiperbolikus párhuzamossági axióma is. Szükségünk van továbbá a távolságfüggvény bevezetésére.

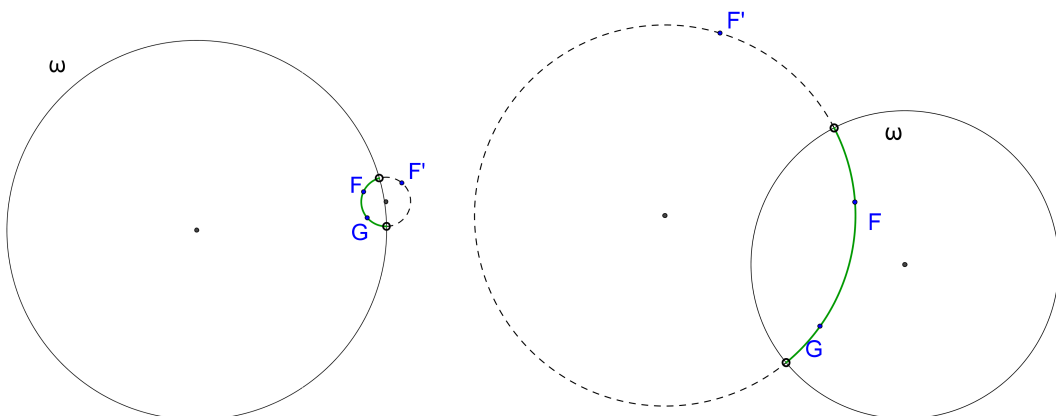
Definíció: Legyenek $P, Q \in int\omega$ pontok. A P és Q pontra illeszkedő ω -t merőlegesen metsző körnek a ω körrel való metszéspontjai legyenek U, V pontok, továbbá legyen

$$d(P, Q) = \left| \ln \left(\frac{UP}{PV} : \frac{UQ}{QV} \right) \right|.$$

Belátható, hogy a fent definiált d függvény megfelelő távolságfüggvény-interpretáció.



3.1. ábra. Egyenesek a Poincaré-féle körmodellben



3.2. ábra. Két pontra (F és G) illeszkedő egyenes

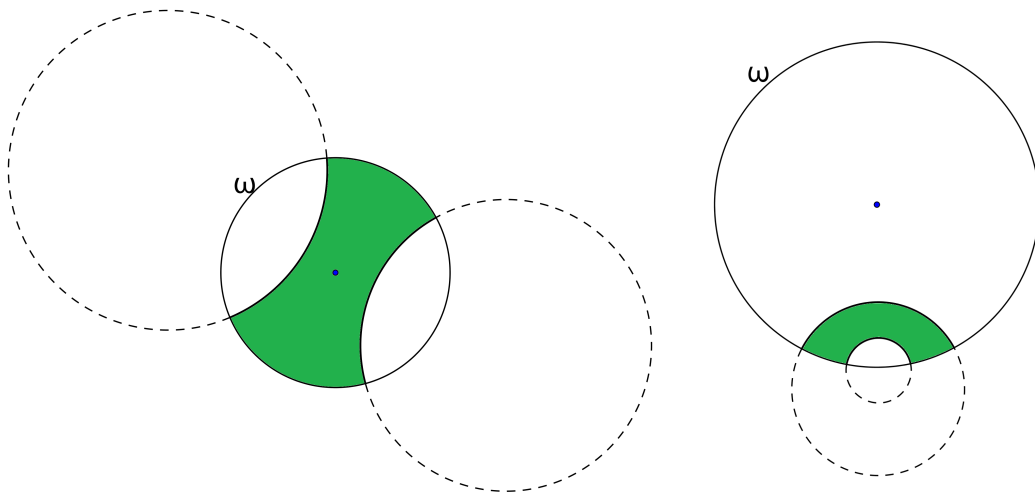
A modellben két egyenes segítségével könnyen meg tudunk jeleníteni egy kapcsolatot két körív között, melynek reprezentálására a 3.3. ábrán látható zölddel színezett terület szolgálhat.

Legyen $S, T \in \omega$, ekkor egyértelműen létezik olyan k E-kör, mely ω -t merőlegesen metszi az S és T pontokban.

Belátható, hogy S és T távolságának csökkenésével csökken a k E-kör sugara is, továbbá k középpontja eleme az $\text{int}\omega \cup \omega$ halmaz komplementerének.

Ebből következően k és O (ω középpontja) távolsága egyre nagyobb a jelzett esetben. A fentiekből látszik, hogy a modell alkalmazásával elkerülhető az egymáshoz közel eső kis ívhosszú körívek által meghatározott kapcsolatok O ponthoz való indokolatlan közelsége, amit a Circosban tapasztalhatunk.

Tehát láthatjuk, hogy a Poincaré-féle körmodell alkalmas lesz arra, hogy kapcsolatokat jelenítsünk meg a kör két része között, így helyettesíteni tudjuk vele a Bézier-görbéket és az azok használatából adódó hibákat. Emellett a modellben lehetőségünk van távolság meghatározására, amit a későbbiekben fel tudunk használni arra, hogy a kapcsolatokat meghatározó egyeneseket az alapján felosszuk.



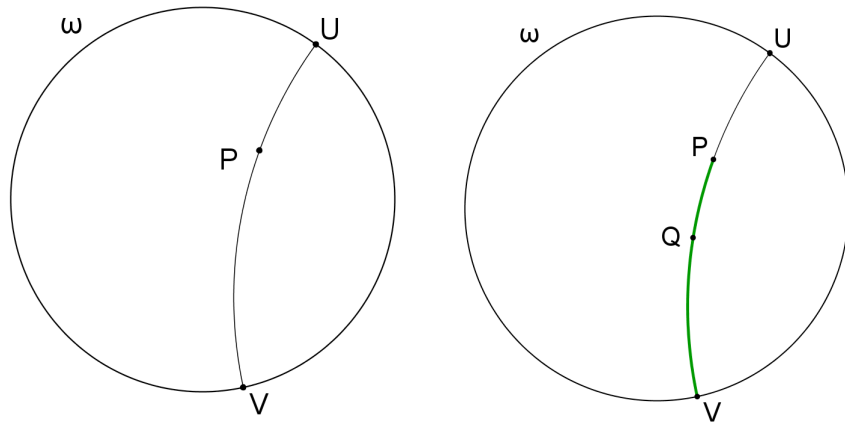
3.3. ábra. Kapcsolatmegjelenítési lehetőség a Poincaré-féle körmodell felhasználásával

3.1.1. Apollóniusz-kör

Terveink között szerepelt, hogy a fentiek alapján létrehozott, kapcsolatokat reprezentáló területeket valamilyen módon felosszuk további vizualizációs lehetőségek érdekében.

Arra gondoltunk, hogy mindenképpen a modellben definiált távolság alapján szeretnénk a mérést végezni, ezzel megtartva a lehetőségét annak, hogy mindvégig a modell geometriájában dolgozhassunk, kihasználva annak előnyös tulajdonságait. A kapcsolatokat meghatározó E-köríveket szeretttük volna egy bizonyos arányban felosztani. Ehhez elengedhetetlen, hogy lássuk, hogyan tudunk egy egyenesen egy adott ponttól megadott távolságra lévő pontot meghatározni.

Tegyük fel, hogy adottak az $U, V \in \omega$ és $P \in int\omega$ pontok, továbbá keressük azt a Q



3.4. ábra. Az U, V, Q és P pontok elhelyezkedése

pontot, melyre a PQ távolság egyenlő egy adott t értékkel. P illeszkedjen az U és V által a fentiek szerint meghatározott egyenesre és tegyük fel, hogy Q a PV modellbeli félegyenesen helyezkedik el. Ekkor igaz a következő:

$$t = -\ln \left(\frac{UP}{PV} \cdot \frac{QU}{VQ} \right).$$

Melyből kis átalakítással kapjuk az

$$e^{-t} = \left(\frac{UP}{PV} \cdot \frac{QU}{VQ} \right)$$

egyenletet.

Legyen $l = \frac{UP}{PV}$ és $f = e^{-t}$ ekkor a fenti egyenlet

$$\frac{f}{l} = \frac{QU}{VQ}$$

alakba hozható.

Látható, hogy olyan Q pontot keresünk, melynek U -tól és V -től mért távolságának aránya ismert. Tudjuk, hogy az ilyen tulajdonságú pontok egy körre illeszkednek, melyet Apollóniusz-körnek nevezünk.

Definíció: Az Apollóniusz-kör azon pontoknak a mértani helye a síkban, amelyeknek két adott ponttól mért távolságaik aránya állandó.

Tehát kaptunk egy arányt az UQ és QV hosszakra vonatkozóan. Ezt felhasználva keressük azt az Apollóniusz-kört, ami az U -t V -vel összekötő (euklideszi értelemben vett) szakaszt $\frac{f}{l}$ arányban osztja fel. A továbbiakban ezt a hányadost jelöljük λ -val. A kör sugara az $r = \left| \frac{\lambda}{(\lambda^2 - 1)} \right| \cdot UV$ képlettel számolható, a kör középpontjának a meghatározása pedig λ ismeretében az U -t és V -t tartalmazó $\mathbb{R}^2$ -beli lineáris sokaság elemei közül könnyen meghatározható. A keresett Q pont ennek a körnek és a már említett, U és V pontok által meghatározott modellbeli egyenesnek a közös pontja lesz.

Azt akartuk elérni, hogy a kapcsolatokon belül további adatokat tudjunk megjeleníteni úgy, hogy azokat ezen értékek alapján arányosan felosztjuk. Mivel a kapcsolatokat modellbeli egyenesek reprezentálják, és egy egyenest képtelenség úgy két részre osztani, hogy a részek hosszainak aránya egy bizonyos szám legyen, ezért ennek a kiküszöbölésére egy az alapkörrel koncentrikus kört vezettünk be, és az egyenesnek ezen a körön belül eső szakaszaival dolgoztunk inntől kezdve, ezek végpontjaiból a fentebb leírtak szerint már könnyen mérhetünk távolságot.

Ha ezeket a szakaszokat fel tudjuk osztani, akkor a kapcsolatokkal is meg tudjuk ezt tenni, mivel két ilyen szakasz által közbezárt terület fogja őket mostantól reprezentálni.

3.2. A matematikai modell használatából fakadó előnyök

Az alábbi előnyöket nyertük a Poincaré-féle körmodell használatával:

- Egy szakaszt a két végpontja egyértelműen meghatározza, nem kell paraméterezéssel foglalkozni.
- A kapcsolatok nem fognak egy pontban összetorlódni, így nem fogják elveszíteni a jelentésüket és átláthatatlanná tenni az ábrát.
- A modellből adódóan nem fordulhat elő, hogy metszéspontja legyen két olyan egyenesnek, amelyek együtt egy kapcsolatot alkotnak.

- A modellből következően az egymáshoz közeli körgyűrűcikkeket összekötő kapcsolatok nem fognak a kör közepe felé benyúlni.
- Lehetőségünk van az egyenesek közötti metszéspontok meghatározására.
- A kapcsolatokat a hosszuk, illetve a polárkoordinátáik alapján arányosan fel tudjuk osztani.

Tehát sikerült egy olyan modellt találnunk, ami megoldja a kapcsolatok problémáját, és emellett nem vezet be újabbakat, továbbá jó alappal szolgál a további fejlesztésekhez, mivel még több olyan tulajdonsága is van, amit nem vizsgáltunk meg a vizualizáció szempontjából.

3.3. Kapcsolatok rendezése

Bizonyos Circossal készült képeken a kapcsolatok nehezen áttekinthetőek voltak. Ez alatt azt értjük, hogy a diagramban túl sok kapcsolat metszette egymást, így az átláthatatlan lett. Ez leginkább magas kapcsolatszám mellett volt megfigyelhető. Innen vettük az ötletet, hogy egy előzetes rendezést végezzünk a szakaszokon és a kapcsolatokon. Ha nem fontos ezeknek az egymáshoz viszonyított sorrendje, akkor a rendezéssel javítani tudnánk a diagram átláthatóságán a kapcsolatok metszésszámának redukálásával. A linkek metszésének vizsgálatához minden adott volt, mivel két körre könnyen meghatározható, hogy metszik-e egymást, és ha igen, akkor ez melyik pontban történik. Mivel a *Circular layout* technikát alkalmazva a kapcsolatok metszésének redukálása egy *NP-nehéz* probléma [4], ezért először csak egy közelítő megoldás megtalálását tűztük ki célul, amely valamilyen heurisztikán alapul. A heurisztika a megjelenítés sajátosságait veszi alapul és emellett a Poincaré-féle körmodell tulajdonságait használja fel. A 3.5. ábrán láthatjuk, hogy mennyivel átláthatóbb vizualizációt érhetünk el úgy, ha rendezzük a kapcsolatokat, és a szakaszokat.

A rendezés folyamata

Két dolgot veszünk jelenleg figyelembe egy szakasz elhelyezkedésének meghatározásánál. Az egyik, hogy milyen sok kapcsolatot tartalmaz, a másik pedig az, hogy mennyi olyan kapcsolattal rendelkezik, amely már valamelyik a rendezés során elhelyezett szakaszhoz kapcsolódik.

A szakaszok sorba rendezését az alábbiak szerint végezzük, legelsőnek a legtöbb kapcsolatot tartalmazó szakaszt választjuk, őt fogjuk az első elemnek tekinteni a sorban. Ezután a

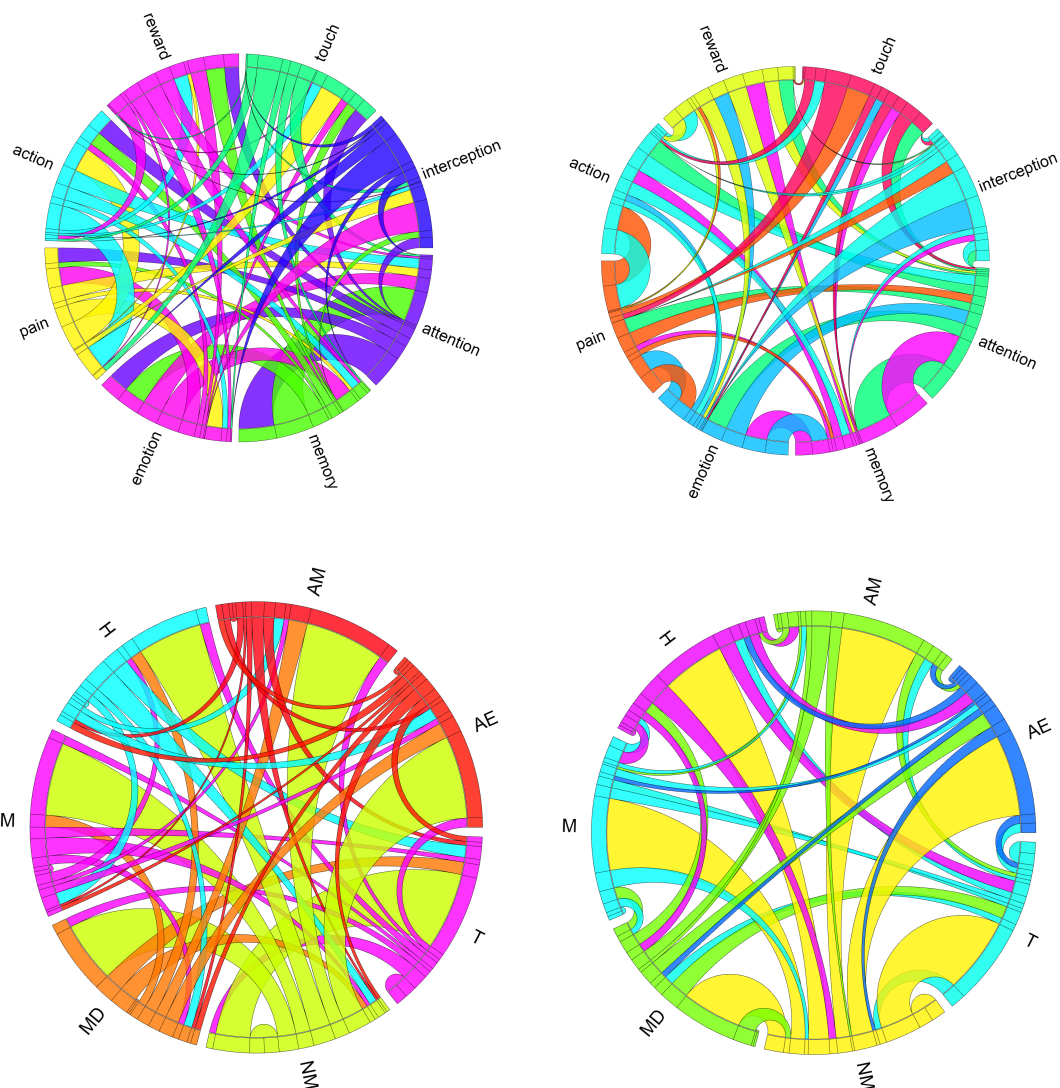
következő elemnek megpróbálunk olyan minimális kapcsolatszámmal rendelkező, a sorban még nem szereplő szakaszt találni, amely kapcsolatban van a sorban legutoljára elhelyezkedővel. Több ilyen szakasz megtalálása esetén, azt fogjuk közülük a következő elemnek tekinteni a sorban, amelyik a legkevesebb olyan kapcsolattal rendelkezik, mely már egy sorba rendezett szakaszra mutat. Ha az első lépésben nem találtunk olyan szakaszt, ami megfelelt volna a kritériumainknak, akkor a sorban következő szakasznak azt a szakaszt fogjuk választani, amelyik a legkevesebb olyan kapcsolattal rendelkezik, amelyik egy másik sorba rendezett kapcsolatra mutat, és minimális a kapcsolatainak a száma. Az egyes szakaszokban lévő kapcsolatok elhelyezkedését a szakaszok sorrendje alapján határozzuk meg. Ezzel az eljárással egy sok szempontból kielégítő rendezéshez juthatunk.

3.4. Kapcsolatok felosztása

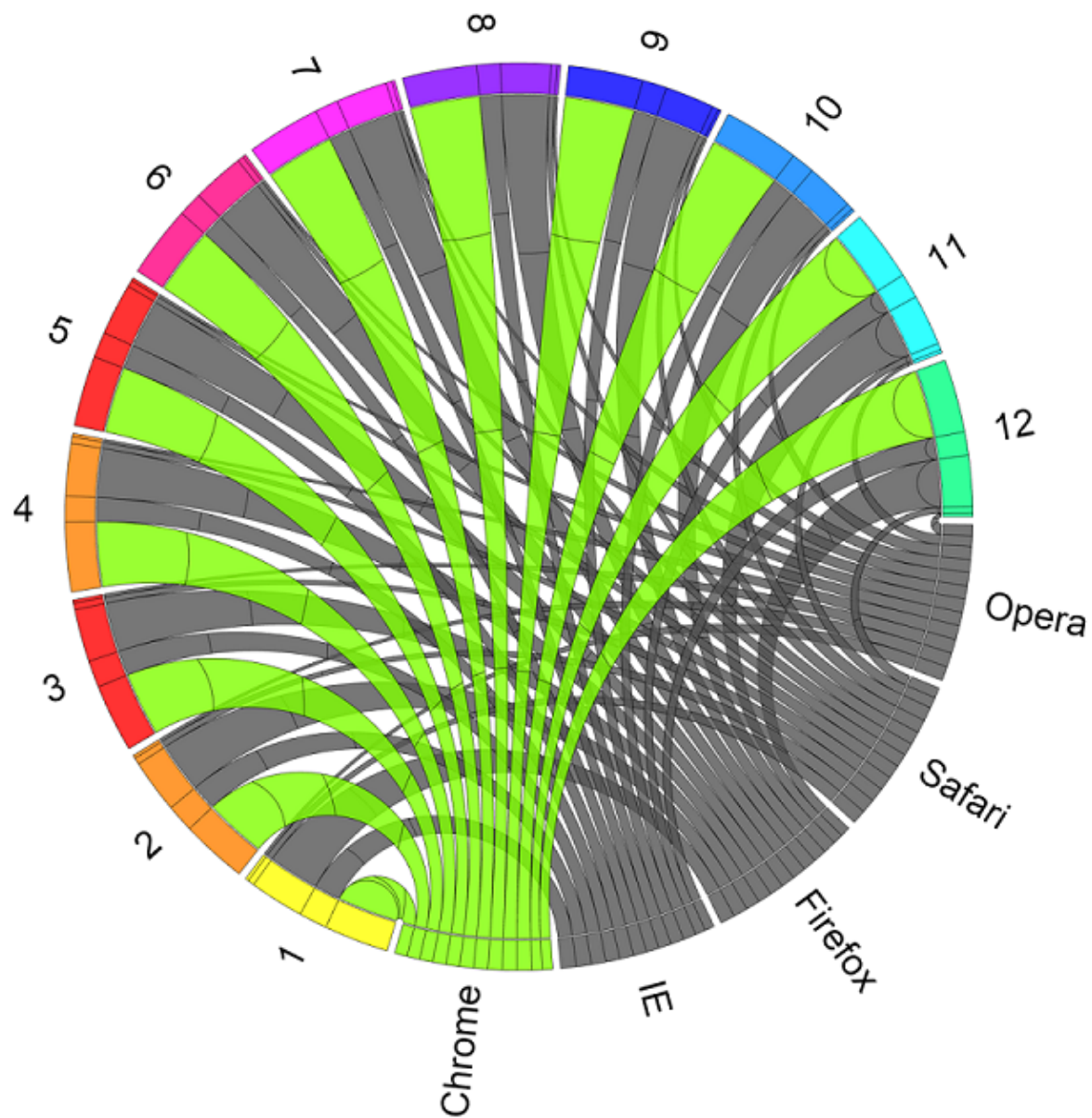
Azzal, hogy a kapcsolatokat fel tudjuk osztani, lehetőséget nyertünk arra, hogy több információt tudjunk közölni a vizualizációval. Ez abban nyilvánul meg, hogy így több adatot tudunk egyszerre megjeleníteni az ábrán, ami segíti a következtetések levonását, a minták felismerését. Lényegében a felosztással a kétdimenziós mátrixunkat háromdimenziós mátrixra tudtuk kiterjeszteni, és így képesek vagyunk történelmi adatokat is belevinni a megjelenítésbe (lásd 3.6. ábra).

3.4.1. A felosztás menete

A kapcsolatok felosztását kétféle módon tudjuk elvégezni. Az egyik módszer amikor a kapcsolatokat alkotó szakaszok hiperbolikus síkon mért hosszát vesszük alapul. A kapcsolatban elhelyezkedő adatoknak meghatározzuk az összegét, majd ugyanezen adatokhoz kiszámoljuk az arányukat az összeghez képest. Ezzel megkaptuk az arányokat, amiket már csak meg kell szorozni a kapcsolat hosszával, és így megkaptuk a felosztások hosszát. Ezek után az Appolloniusz-kör használatának a segítségével a felosztások kiszámított hosszainak alapján fel tudjuk osztani a kapcsolatot. Ennek a módszernek a segítségével a további felosztás is lehetséges lehet a későbbiekben. A másik módszer, amikor a kapcsolatot alkotó szakaszoknak a két végpontja közötti szöget határozzuk meg az euklideszi síkon. A felosztás meghatározása az előbb leírtaktól annyiban különbözik, hogy itt a kapcsolat hossza helyett az előbb a kapcsolatra vonatkozóan megkapott szöget használjuk fel. Tehát a kapcsolatokat mind euklideszi, mind hiperbolikus síkon fel tudjuk osztani.



3.5. ábra. Rendezés nélkül (bal oldalon) és rendezéssel (jobb oldalon) elkészített vizualizációk. Fent az agy működésével kapcsolatos adatok [1], lent pedig a periódusos rendszer elemeinek csoportjai közötti összefüggések [15] vizualizációja látható.

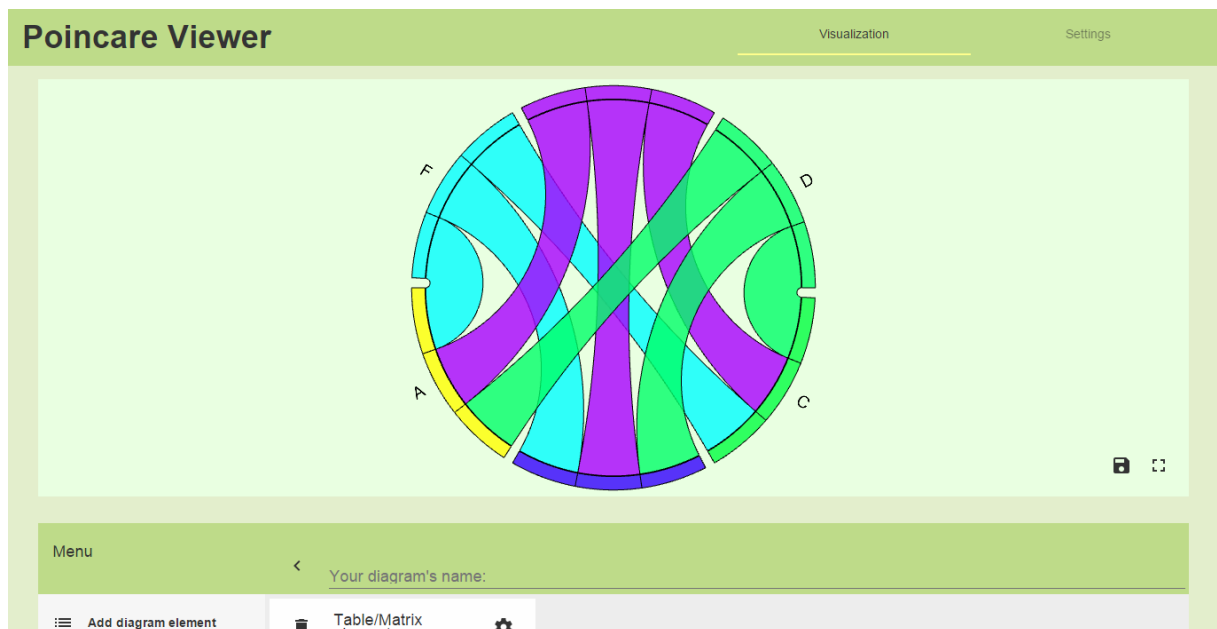


3.6. ábra. A böngészőpiac alakulásának adatai havi bontásban, évenként levetítve (kapcsolatok felosztása a kapcsolatok végpontjai között mért szögek alapján) [24], a Poincaré Viewer alkalmazásunkkal ábrázolva

4. fejezet

Az alkalmazásunk

A Poincaré-féle körmodellt implementálva a kapcsolatok megadására készítettünk egy alkalmazást (lásd 4.1. ábra), amivel az elért eredményeket, valamint a modell által adott lehetőségeket akarjuk szemléltetni. A fejlesztés alatt mindig szem előtt tartottuk, hogy az elkészült szoftver jól összehasonlítható legyen a Circossal. Ennek a szoftvernek a Poincaré Viewer fantázianevet adtuk.



4.1. ábra. A Poincaré Viewer alkalmazás felhasználói felülete

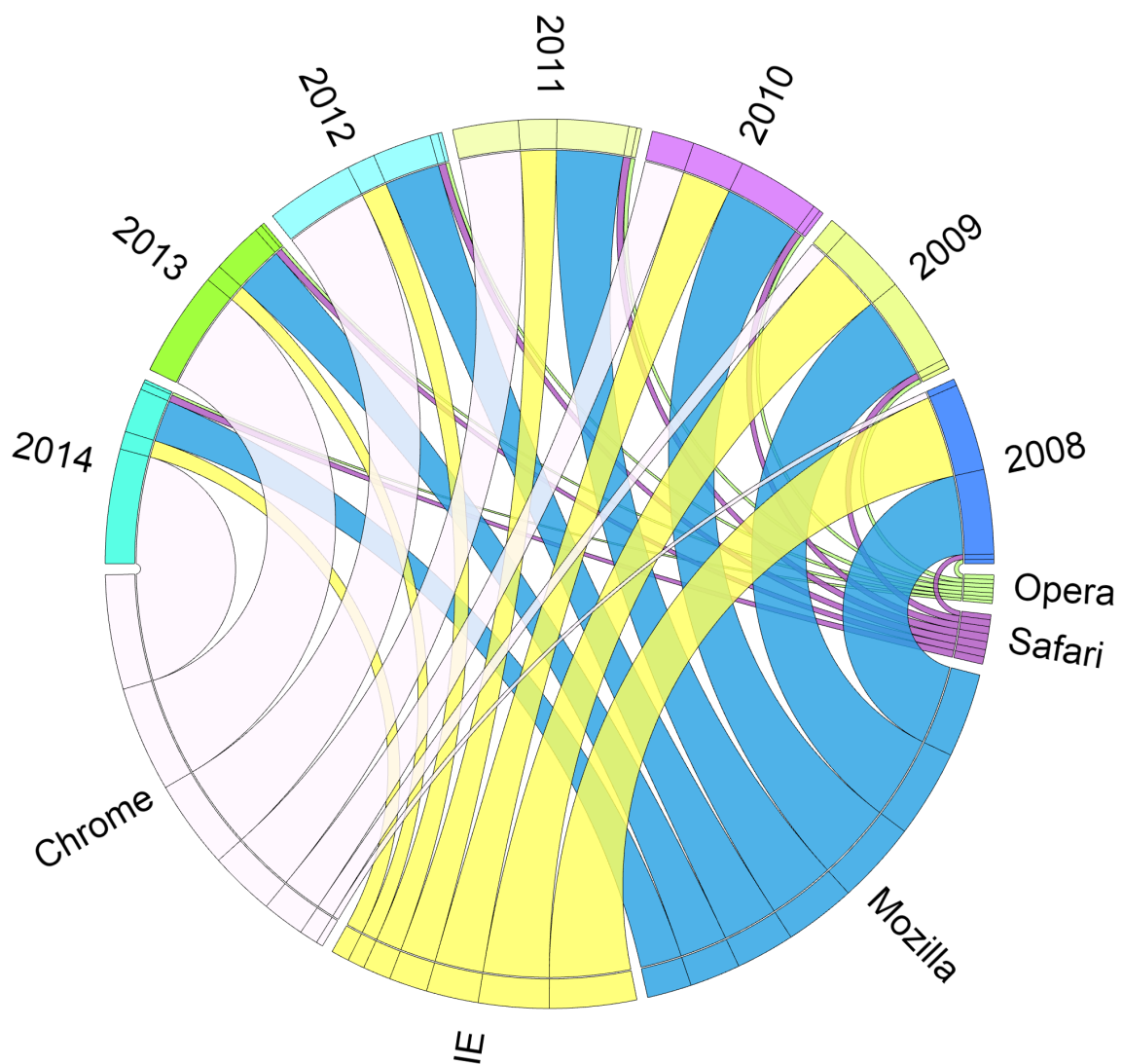
4.1. Irány a web

Azt szerettük volna, hogy a kész szoftvert minél szélesebb körben tudják használni a felhasználók, akár professzionális, akár egyszerű vizualizációk elkészítéséhez. Úgy láttuk, hogy ezt legkönnyebben egy webalkalmazáson keresztül érhetjük el, mivel így megkaphattuk a multiplatformon való működést, és a HTML-nek köszönhetően egyszerűen elkészíthető grafikus felület is a rendelkezésünkre állt. Ebben az is segített minket, hogy a legtöbb böngésző már majdnem teljes támogatást nyújt a HTML 5 használatához, amit most már hivatalos ajánlásként minősít a W3C [23]. Interaktívan szerkeszthető ábrákat akartunk a felhasználóknak nyújtani, amihez pontosan megfelelt a szintén már a legtöbb böngésző által támogatott WebGL. Mivel a modellben a kapcsolatok felépítéséhez használt egyenesek meghatározása nem nagy számításigényű probléma, ezért mindenképpen csak egy kliensoldali alkalmazásban gondolkodtunk, a Circos Table Viewerrel ellentétben.

4.2. Technológiai újdonságok

Mikor kiválasztottuk azt, hogy milyen nyelven írjuk a programunkat és azt, hogy mivel jelenítsük meg a diagramunkat, továbbá, hogy mit használjunk a GUI felépítéséhez, mindvégig azt tartottuk szem előtt, hogy a jelenleg elérhető legújabb és legígéretesebb megoldásokat használjuk. Ezzel nagy kihívás elé állítottuk magunkat, viszont meg tudtuk mutatni az alkalmazásunkon keresztül, hogy mit is tudnak ezek az új megoldások és esetlegesen miért jobbak a versenytársaiknál. Az esetek többségében a kihívást az jelentette, hogy a választott eszközök közül még néhány most is fejlesztés alatt van, így kompatibilitási problémákkal kellett szembenéznünk.

A Poincaré viewert a Google által fejlesztett, a JavaScript leváltására szánt Dart szkript nyelven írtuk. A nyelvnek számos előnye van a JavaScripttel szemben, viszont hátránya, hogy jelenleg csak Dartiumban, a Chromium böngésző egy változatában van integrálva a DartVM, ami ahhoz szükséges, hogy a Dartban íródott kódokat futtatni tudjuk. Ahhoz, hogy más böngészőben is használni lehessen a Dartban megírt szoftvereket, át kell fordítani azokat JavaScriptre. Az okok, melyek miatt az imént említett hátrány ellenére mégis a Dart programozási nyelv mellett tettük le a voksunkat, azok voltak, hogy a Google bejelentette, hogy a Chrome böngészőbe is bekerül a DartVM, illetve a Chrome böngészőnek van a legnagyobb részesedése jelenleg a böngészők piacán [24], amit az 4.2. ábrán kitűnően láthatunk. Ezáltal hamarosan nagyobb célközönség tudja kihasználni a



4.2. ábra. A böngészőpiac alakulásának adatai 2008-ig visszamenőleg [24], a Poincaré Viewer alkalmazásunkkal ábrázolva

Dart nyelv gyorsaságát a JavaScripttel szemben, és mivel a JavaScriptre fordított Dart kód sebessége megközelíti a JavaScript sebességét, így a más böngészőt használók sem szenvednek nagy hátrányt.

A felülethez szintén egy a Google által menedzselte projektet választottunk, a *Polymert*, amely webkomponenseken alapul. Ez jelenleg még gyerekcipőben jár, csak előzetes fejlesztői állapotban (developer preview) van. A projektben megtalálhatók az Android következő verziójának a Lollipopnak a design elemei is, amelyek közül az alkalmazásunkban többet is felhasználtunk.

A szoftverrel készült megjelenítés megvalósítására több lehetőségünk lett volna, mi mégis a WebGL mellett döntöttünk, mivel rengeteg potenciális fejlesztési irányba tudunk majd vele továbblépni, így a későbbiekben nem lesz megkötve a kezünk azzal kapcsolatban, hogy merre haladjunk a kutatásunk során. Ezen kívül egyre több alkalmazást készítenek el WebGL segítségével, tehát egy aktuálisan széles körben használt grafikai API-ról beszélhetünk.

4.3. A látható előny

A modell használatából származó eredmény a kapcsolatok megjelenítésénél nagyszerűen látható. Egyrészt a Circosnál említett problémák nem merülnek fel, másrészt a kapcsolat végpontjainak távolsága meghatározza azt, hogy a link mennyire nyúlik be a kör középpontja felé, ezzel elérve azt, hogy a kör közepe nem lesz túlzsúfolt. Az ábránkon a kapcsolatok körvonalai különböző sugarú és középpontú körökhöz tartoznak, így ezeknek a görbülete azok egésze mentén megegyezik.

Az alkalmazás felületén könnyű eligazodni, pillanatok alatt lehet a segítségével vizualizációt készíteni, akár egy fájlból is, ugyanúgy, mint a Circos Table Viewer alkalmazásnál, vagy a felületen összeállított táblázatból. A kapcsolatok rendezésére valós időben van lehetőség, emellett a diagramot bármilyen irányba lehet mozgatni, jobbra és balra forgatni, illetve kicsinyíteni, nagyítani.

A kapcsolatok automatikus rendezésével el tudtuk érni, hogy egy olyan sorrendet kapjunk, amit már csak kis mértékben, vagy egyáltalán nem kell módosítani, ezzel is csökkentve a vizualizáció testreszabásával töltött időt. Ezen felül a legfontosabb, hogy egy átláthatóbb vizualizációt tudunk nyújtani a felhasználóknak.

Végül pedig a szoftverrel lehetőségünk van nem csak táblázatos adatokból vizualizációt készíteni. Így sokkal szélesebb lett a skálája azoknak az adatoknak, amiket az alkalmazásunkkal meg tudunk jeleníteni. A módszer alapelve, hogy megadjuk mit szeretnénk összekapcsolni, s hogy a megadott elemek között milyen kapcsolatokat akarunk látni. Ezzel elértük azt, hogy nem csak számok alapján tudunk információk között összefüggéseket megjeleníteni.

5. fejezet

A saját alkalmazásunk és a Circos összehasonlítása

Ebben a részben egyszerre vesszük górcső alá a két programot. Azzal, hogy a két alkalmazást egymással szembe állítottuk és tüzetesen megvizsgáltuk, a fent ismertetett eredmények még látványosabb szemléltetését kaptuk. A vizsgálódást több oldalról is elvégeztük. Először a hasonlóságokat derítettük fel, majd pedig a különbségeket. Itt elsősorban a funkcionalitást vizsgáltuk. Ezek után a programok kimeneteit kezdtük el elemezni olyan módon, hogy mindkét szoftverrel ugyanarra az adatsorra elkészítettük a megjelenítést, majd a kapott eredményeket egymás mellé rakva összehasonlítottuk azokat. A képek generálást elvégeztük a Circos Table Viewer webalkalmazással, illetve a Circos programmal is. Több okból tartottuk szükségesnek, hogy mindkét programmal elkészítsük ugyanazt a vizualizációt. Először is mivel több cikkben is csak az online alkalmazást használva készítettek a kutatók vizualizációt. Másodszor a Circos lehetőséget nyújt számunkra a megjelenítés módosítására. A Circos Table Viewerre a továbbiakban a CTV, míg a saját alkalmazásunkra PV rövidítést fogjuk használni.

5.1. Hasonlóságok

Mindkét alkalmazás ugyanazon technikát használja az adatok vizualizációjának elkészítésére, a *Circular layout* technikát. A megjeleníthető adatokban a PTV és a CTV között nincsenek nagy eltérések, mivel a programokkal olyan táblázatokat van lehetőségünk vizualizálni, amelyek elemei számok. A futtatás után kapott eredmény is mindkét szoftver esetében hasonló, mivel a táblázat sorai és oszlopai közötti kapcsolatokat ábrázolják. A cél mindkét esetben azonos, mégpedig az, hogy az ábrán látható kapcsolatokról továb-

bi összefüggéseket tudjunk levonni. A bemeneti fájlok között is van hasonlóság, addig a pontig, amíg csak magát a táblázat részét képező adatokat tartalmazza.

5.2. Különbségek

Ha a különbségeket vizsgáljuk, akkor már több dolgot figyelhetünk meg. Az első eltérést a megjelenítésben fedezhetjük fel, azon belül is az ábrán megjelenő linkekben. A linkek alakja eltérő a két szoftverben. Ennek az okát a kapcsolatok meghatározásához használt különböző modellekben kell keresni. A CTV Bézier-görbék segítségével írja le a kapcsolatokat, míg a PV-ben ezek leírására a Poincaré-féle körmodell egyeneseit használjuk. A CTV nem jeleníti meg a pontos értékeket a kapcsolatokhoz, hanem csak egy számsort, amiről a körülbelüli értékük olvasható le. Ezen felül külső sávokban további százalékos megoszlási adatokat láthatunk. A PV-ben a kapcsolatokhoz tartozó táblázatbeli adatok jelennek meg a kör szélén a linkek végénél. A bemeneti adatok megadására a PV-ben grafikus felületen is lehetőség van. A felhasználói felületnél maradva feltűnik, hogy a CTV alkalmazásban nincs lehetőség interakciók végzésére az elkészült kimeneten, míg a PV-ben igen. A kapcsolatok és szakaszok sorrendjét valós időben tudjuk módosítani, valamint lehetőség van a diagramot nagyítani, kicsinyíteni, forgatni és mozgatni. A kimenet tekintetében a CTV lehetőséget ad arra, hogy a konfigurációs beállításokat is letöltsük.

5.3. Összehasonlítás képekben

Ebben a részben a különböző adatokból az alkalmazásokkal készült vizualizációkat egymás mellé állítottuk, hogy összehasonlíthassuk őket. A képeket az összehasonlításhoz az alábbiak szerint csoportosítottuk. Három táblázathoz készítettünk összehasonlítást, melyeket úgy rendeztük, hogy az ugyan azt az adatot ábrázolók egy oldalon helyezkedjenek el. Az első két kép az oldalon minden esetben a Circos asztali alkalmazásnak és a Circos Table Viewer programnak a görbeparaméterek módosítását mellőző kimenetei lesznek, amikre a továbbiakban mint alapértelmezett kimenetek hivatkozunk. Ezeket követik a Circossal különböző paraméterbeállítások mellett elkészült képek, majd pedig a PV-vel, a saját alkalmazásunkkal készült vizualizáció fog következni. Az adatok, melyek alapján a megjelenítést végeztük, az A. függelékben találhatók meg. A paraméterek rövidítésére a következőket alkalmazzuk: $\text{bezier_radius} = \text{br}$, $\text{bezier_radius_purity} = \text{brp}$, $\text{crest} = \text{c}$.

Először a böngészőpiac alakulását [24] leíró táblázatot (lásd A.1. táblázat) vizualizáltuk (lásd 5.1. – 5.3. ábra).

A képekről látszik, hogy már a Circos és a CTV alapértelmezett kimenetei (lásd 5.1. ábra) is produkálják az általunk felismert hibákat, és a paraméterek változtatásával sem sikerült elérni jobb eredményt, egyes esetekben még rosszabb lett (lásd 5.2. ábra jobb oldalán szereplő kép), mivel teljes kapcsolatokat sikerült eltakarnunk. Ellenben a PV-ben ezek a problémák nincsenek jelen (lásd 5.3. ábra).

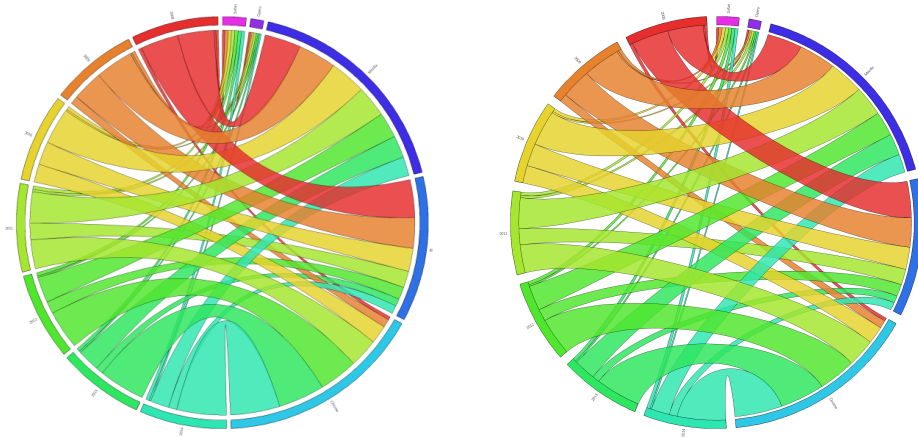
A következő képsorozat (lásd 5.4. – 5.6. ábra) az autómárkák és üzemanyag típusok kapcsolatát (lásd A.2. táblázat) illusztrálja [7].

Itt is hasonló esettel találkoztunk szembe, az alapértelmezett kimenetekben (lásd 5.4. ábra) ismételten jelentkezett a hiba. A paraméterezéssel ebben az esetben lokálisan már sikerült jobb eredményt elérnünk, azonban így globálisan rontottuk el a diagramot. A Poincaré Viewerrel kapcsolatban ismét megjegyezhetjük, hogy nem mutatott semmilyen hibát (lásd 5.6. ábra).

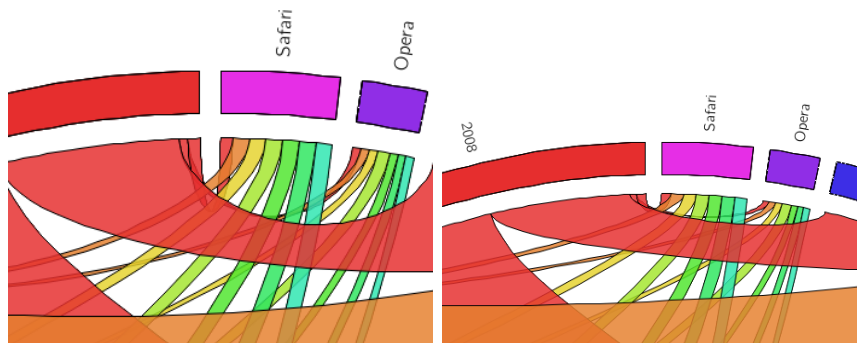
Utolsó sorban pedig egy az agy működésével kapcsolatos adatokat tartalmazó táblázat [1] (lásd A.3. táblázat) vizualizációján keresztül szeretnénk az alkalmazásokat összehasonlítani (lásd 5.7. – 5.9. ábra).

Ebben az esetben is ugyanarra a következtetésekre juthatunk a képek vizsgálata után. A Circos alapbeállítások mellett hibás vizualizációt produkál (lásd 5.7. ábra) és a paraméterek megváltoztatásával sem tudunk javítani a helyzeten (lásd 5.8. ábra). Azonban elmondhatjuk, hogy a saját alkalmazásunk most sem produkált semmilyen a Circosban észrevett problémát.

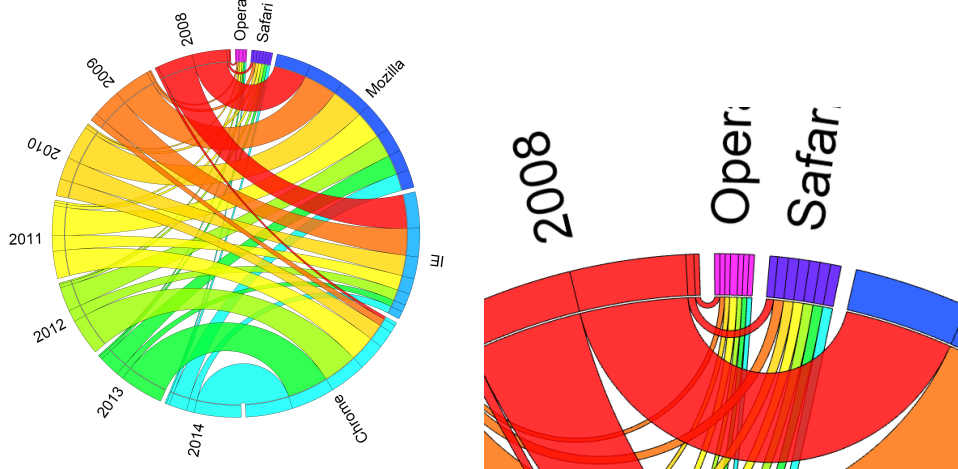
Összességében azt mondhatjuk, a kapcsolatok vizualizációjára adott megoldásunk megfelelően teljesített és egyetlen hibát sem produkált. Ha a Circos és a Circos Table Viewer alapértelmezett kimeneteit tekintjük, akkor látható, hogy nem azonosak az alapbeállítások. A webalkalmazással készült képek sokszor jobb eredményt mutattak, mint az asztali verzió, viszont messze nem megfelelő. A további képek, ahol a CTV-t és a PV-t hasonlítottuk össze, a B. függelékben találhatók.



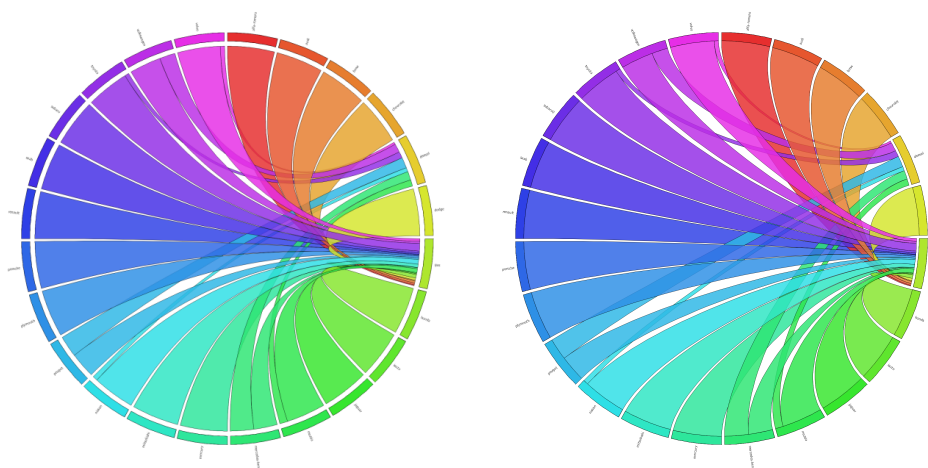
5.1. ábra. A Circos (bal oldalon) és a Circos Table Viewer (jobb oldalon) alapértelmezett kimenetei az A.1. táblázat adataira



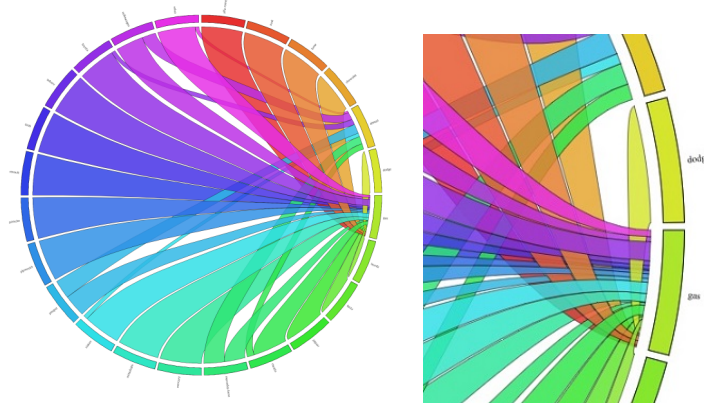
5.2. ábra. A Circos alkalmazással nyert kimenetek az A.1. táblázat alapján. A jobb kép paramétereit (brp: 0.25, br: 0.5), bal kép paramétereit (brp: 0.05, br: 0.2, c: 1)



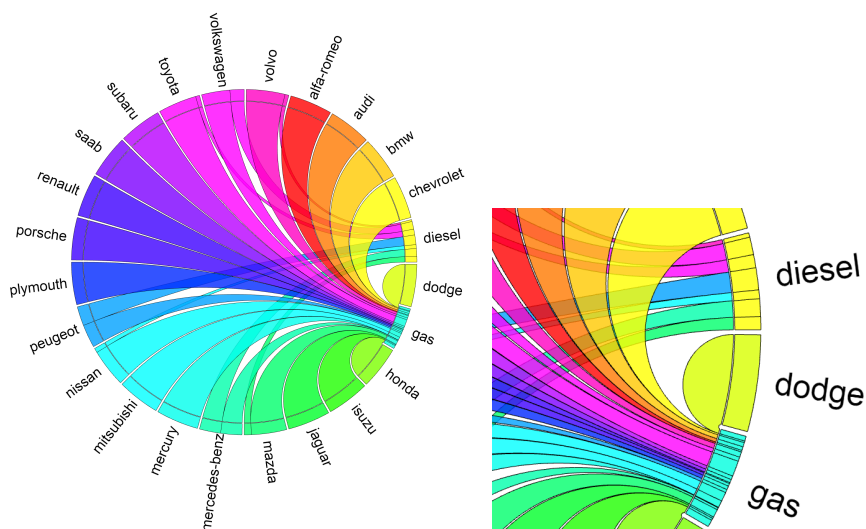
5.3. ábra. A saját szoftverünk (Poincaré Viewer) kimenetei az A.1. táblázat adataira



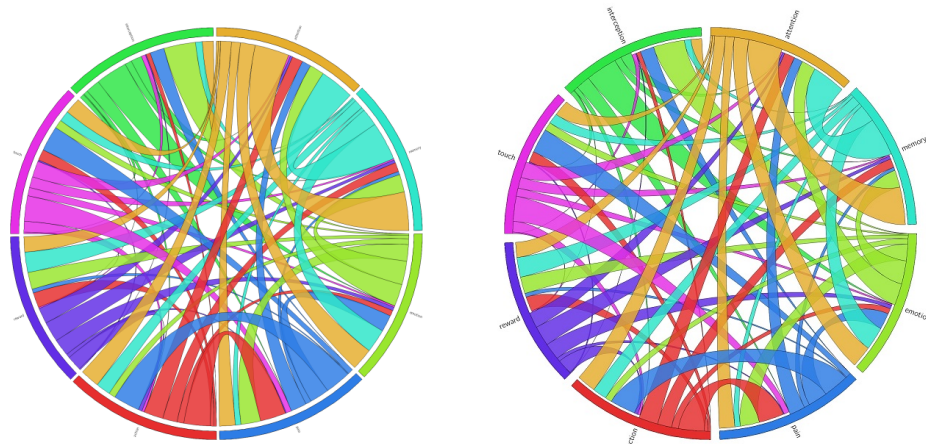
5.4. ábra. A Circos (bal oldalon) és a Circos Table Viewer (jobb oldalon) alapértelmezett kimenetei az A.2. táblázat adataira



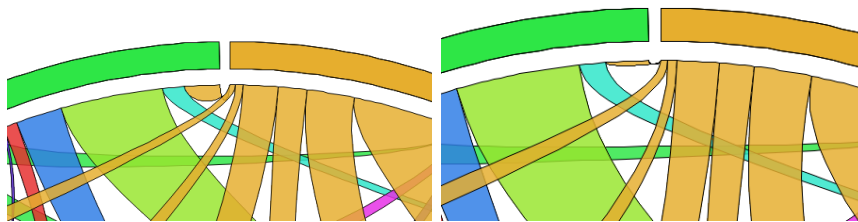
5.5. ábra. A Circos alkalmazással nyert kimenetek az A.2. táblázat alapján. A képek paramétereit (brp: 0.025, br: 0.8, c: 1)



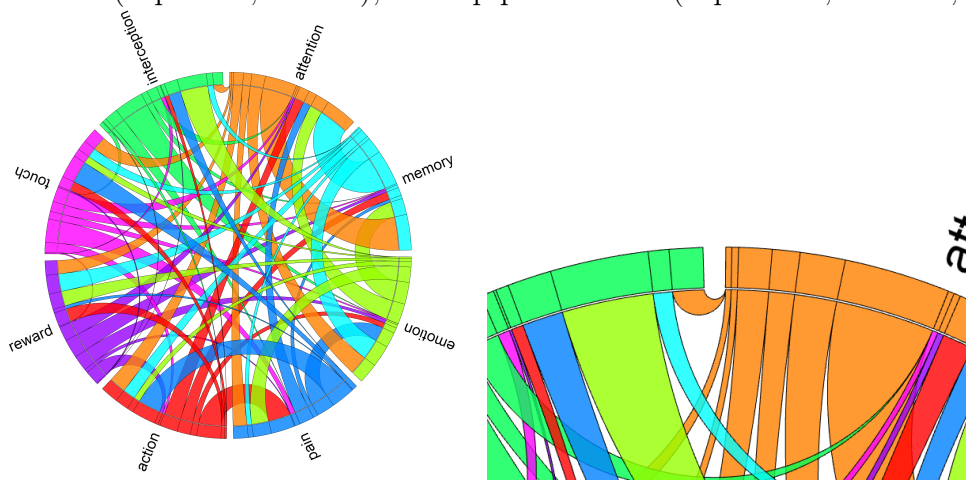
5.6. ábra. A saját szoftverünk (Poincaré Viewer) kimenetei az A.2. táblázat adataira



5.7. ábra. A Circos (bal oldalon) és a Circos Table Viewer (jobb oldalon) alapértelmezett kimenetei az A.3. táblázat adataira



5.8. ábra. A Circos alkalmazással nyert kimenetek az A.3. táblázat alapján. A jobb kép paramétere (brp: 0.05, br: 0.2), bal kép paramétere (brp: 0.025, br: 0.75, c: 1)



5.9. ábra. A saját szoftverünk (Poincaré Viewer) kimenetei az A.3. táblázat adataira

6. fejezet

Összefoglalás

Az adat és az információ napjainkra egyre fontosabb szerepet tölt be az életünkben. Erre egy jó példa, hogy minden oldalról információval bombáznak minket, például: tv, internet, hírlapok, folyóiratok. Ezek az információk már számunkra értelmezhető módon vannak találva, de ahhoz ezeket valakinek valamilyen módon el kellett készíteni. A nyers adatok érthető formába történő transzformálása nem egyszerű feladat, főleg ha nagy mennyiségű adatról van szó. Ez a feladat valamilyen informatikai megoldást rejt a háttérben, mégpedig az adatvizualizációt. A dolgozatban ebben a témakörben elért eredményeink kerültek bemutatásra.

Feltett szándékunk volt már a kutatások kezdetén, hogy mindig aktuális problémákkal foglalkozzunk, illetve azokra olyan megoldást nyújtsunk, amelyek jelenleg új és fejlett módszernek, technológiának, illetve technikának számítanak. Ezek alapján választottuk a genomi adatok vizualizációját is.

Kezdetnek meg akartuk ismerni, hogy az adatvizualizációt tekintve milyen megoldások láttak eddig napvilágot az általunk választott területen, ezért tüzetesebben is megvizsgáltuk a genomi adatok vizualizációjával foglalkozó alkalmazásokat. A vizsgálódásunk eredménye alapján egy program kiemelkedett a többi közül, ez pedig a Circos volt, így a továbbiakban vele folytattuk a kutatást. Több problémát, illetve hiányosságot is felfedeztünk benne. Mivel a Circossal lehetőségünk van nem genomi adatok vizualizációjára, és azokon is észleltük a problémákat, így a kezdeti feltevésünket, hogy genomi adatok vizualizációját valósítsuk meg, feláldoztuk annak érdekében, hogy a Circosban megtalált hibákra olyan megoldást találjunk, továbbá a felfedezett hiányosságokat olyan új ötletekkel funkciókkal orvosoljuk, hogy vizualizációban túlszárnyaljuk a Circost. Ezeken felül egy olyan felhasználói felület kialakítását terveztük, ami megfelel a mai minimalista design

irányvonalnak, illetve lehetőséget ad a felhasználóknak, hogy kedvük szerint interakciókat végezzenek a diagramon. A fenti törekvéseink elérésének érdekében a genomi adatok helyett egy egyszerűbb adatszerkezettel, a mátrixokkal dolgoztunk tovább.

A vizualizációt céljaink eléréshez teljesen új alapokra helyeztük. Azért is állíthatjuk ezt, mivel elhagytuk a megszokott Descartes-féle koordináta-rendszert és helyette a polárkoordináta rendszert részesítettük előnyben. Így megnyílt az út, hogy a Poincaré-féle körmodell segítségével ábrázoljuk a kapcsolatokat az ábráinkon. Emellett a modellbeli szakaszokat arányosan fel tudjuk osztani az Appollóniusz-kör, illetve a polárkoordináták segítségével. Így egy olyan vizualizációt tudtunk összeállítani, ami megfelelő volt a céljaink eléréséhez.

A fenti összehasonlítás remekül illusztrálja az elért eredményeinket, így ezek alapján szeretnénk azokat összefoglalni.

A kapcsolatok megjelenítésében több hibát is felfedeztünk, ezek egyike volt a kapcsolatok „önmetszése” (lásd 5.1, 5.4. és a 5.7. ábrák), egy másik, hogy sok esetben átláthatatlanul helyezkednek el, amivel gátolják az ábra megértését (lásd 5.2. és 5.5. ábrák), valamint a vékony kapcsolatok sok esetben eltűnnek a kör szélénél (lásd 5.7. ábra). Ezeket a hibákat fent bővebben is kifejtettük, illetve az „önmetszésre” megpróbáltunk megoldási javaslatot tenni a Circoson belül, viszont ezek messzemenően nem orvosolták a problémát. Hiányossággként emelném ki, hogy a táblázatos megjelenítésnél nem lehet megadni két adathoz több kapcsolatot és nem lehet explicit megmondani a linkek sorrendjét. Az ábra módosítása nehézkes és időigényes folyamat, és a legtöbb esetben nem tudjuk vele elérni a kívánt eredményt.

Mikor a felhasználói felületek összehasonlítására került a sor, akkor megfigyelhettük, hogy a CTV alig nyújt lehetőséget, hogy interaktívan módosítsuk az elkészült vizualizációinkat. A megjelenítést csak szöveges fájlokon és parancssori argumentumokon keresztül tudjuk vezérelni.

A kapcsolatok problémájára adott megoldásunk, hogy új modellt alkalmazzunk a linkek megjelenítésére, egyszerre oldotta meg azokat (lásd 5.3., 5.6. és a 5.9. ábrák), illetve több olyan lehetőséget szolgáltatott számunkra, amelyeknek a segítségével jobb vizualizációt tudtunk létrehozni. A hiányosságokra a grafikus felhasználói felület szolgáltatott egyszerű és kézenfekvő megoldást. Nem utolsó sorban pedig a kapcsolatok metszésének redukálására egy heurisztikus rendezést adtunk meg, aminek a segítségével sokkal átláthatóbb vizualizációk készíthetők (lásd 3.5. ábra).

A további kutatás és munka irányvonalának a kapcsolatok metszésszámának redukálására használt heurisztika finomítását tervezzük, amit egy megoldást kereső rendszerrel akarunk kiegészíteni. Ezen felül pedig a modell további tulajdonságainak kihasználásával a vizualizációt térben is meg akarjuk valósítani.

Ezek alapján biztosan állíthatjuk, hogy sikeresen elértük a célunkat és egy olyan alkalmazást hozunk létre, amivel bárki egyszerűen vizualizálni tudja az adatait, ráadásul ezt bármilyen platformon megteheti, amely rendelkezik WebGL támogatással.

Legvégül szeretném összefoglalni az általam végzett munkát a kutatásban.

- A genomi adatok vizualizációjának területén végzett kutatómunka.
- A Circos hiányosságainak feltérképezése és azokra megoldás keresése.
- A témavezetőmmel közösen megtalált matematikai modellt implementálva egy alkalmazás elkészítése.
- A diagramban szereplő kapcsolatok metszésszámának redukálása egy heurisztikus módszer segítségével.
- Új kapcsolat alapú vizualizációs módszer kidolgozása.

Irodalomjegyzék

- [1] Franco Cauda–Diana M-E. Torta–Katuscia Sacco–Elisabetta Geda–Federico D’Agata–Tommaso Costa–Sergio Duca–Giuliano Geminiani–Martina Amanzio: Shared “core” areas between the pain and other task-related networks. *PloS one*, 7. évf. (2012) 8. sz., e41929. p.
- [2] Mei C. Chuah–Stephen G. Eick: Information rich glyphs for software management data. *Computer Graphics and Applications, IEEE*, 18. évf. (1998) 4. sz., 24–29. p.
- [3] Andy Cockburn–Bruce McKenzie: An evaluation of cone trees. In *People and Computers XIV—Usability or Else! 2000*, Springer, 425–436. p.
- [4] Emden R. Gansner–Yehuda Koren: Improved circular layouts. In *Graph Drawing* (konferenciaanyag). 2007, Springer, 386–398. p.
- [5] Papp György: Nagyméretű háromdimenziós mátrixok vizualizációja hiperbolikus geometriai alapokon. Debreceni Egyetem Informatikai Kar, 2014. szakdolgozat.
- [6] Brian Johnson–Ben Shneiderman: Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Visualization, 1991. Visualization’91, Proceedings., IEEE Conference on* (konferenciaanyag). 1991, IEEE, 284–291. p.
- [7] Fran Juan: Three charts to explore car types | ICTsophy. <http://ictsophy.wordpress.com/2011/08/12/three-charts/>, Elérés ideje: 2014. november 10.
- [8] Daniel A. Keim–Florian Mansmann–Jörn Schneidewind–Tobias Schreck: Monitoring network traffic with radial traffic analyzer. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On* (konferenciaanyag). 2006, IEEE, 123–128. p.
- [9] Daniel A. Keim–Jörn Schneidewind–Mike Sips: CircleView: a new approach for visualizing time-related multidimensional data sets. In *Proceedings of the working conference on Advanced visual interfaces* (konferenciaanyag). 2004, ACM, 179–182. p.

- [10] Martin Krzywinski–Jacqueline Schein–İnanç Birol–Joseph Connors–Randy Gascoyne–Doug Horsman–Steven J Jones–Marco A Marra: Circos: an information aesthetic for comparative genomics. *Genome research*, 19. évf. (2009) 9. sz., 1639–1645. p.
- [11] John Lamping–Ramana Rao–Peter Pirolli: A focus+ context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (konferenciaanyag). 1995, ACM Press/Addison-Wesley Publishing Co., 401–408. p.
- [12] Krzywinski Martin: Circos Tutorials: Links and Relationships - Link geometry // CIRCOS Circular Genome Data Visualization. <http://circos.ca/documentation/tutorials/links/geometry/images>, Elérés ideje: 2014. november 1.
- [13] Krzywinski Martin: Introduction to Circos, Features and Uses // CIRCOS Circular Genome Data Visualization. <http://circos.ca/>, Elérés ideje: 2014. november 2.
- [14] Krzywinski Martin: Circos Table Viewer - Circular Visualization of Tabular Data // with Circos - Circular Genome Data Visualization. <http://mkweb.bcgsc.ca/tableviewer/>, Elérés ideje: 2014. november 3.
- [15] Krzywinski Martin: Articles // CIRCOS Circular Genome Data Visualization. http://circos.ca/presentations/articles/vis_tables1/, Elérés ideje: 2014. november 5.
- [16] Krzywinski Martin: Scientific Literature Images Created with Circos // CIRCOS Circular Genome Data Visualization. http://www.circos.ca/images/scientific_literature/, Elérés ideje: 2014. november 5.
- [17] Daniela Omodei–Danilo Licastro–Francesco Salvatore–Seth D Crosby–Luigi Fontana: Serum from humans on long-term calorie restriction enhances stress resistance in cell culture. *Aging (Albany NY)*, 5. évf. (2013) 8. sz., 599. p.
- [18] Manojit Sarkar–Marc H Brown: Graphical fisheye views. *Communications of the ACM*, 37. évf. (1994) 12. sz., 73–83. p.
- [19] SURFsara: SURFsara Projects. <https://surfsara.nl/services/visualization-projects>, Elérés ideje: 2014. november 29.

- [20] Spatial Sustain: NASA Updates Eyes on Earth Visualization Site | Spatial Sustain. <http://www.sensysmag.com/spatialsustain/nasa-updates-eyes-on-earth-visualization-site.html>, Elérés ideje: 2014. november 29.
- [21] Roberto Tamassia: Handbook of graph drawing and visualization. *AMC*, 10. évf. (2011), 12. p.
- [22] Telerik: Getting Started With PIE! (And Other DataViz Charts). <http://blogs.telerik.com/kendoui/posts/13-11-14/getting-started-with-pie!-%28and-other-dataviz-charts%29>, Elérés ideje: 2014. november 29.
- [23] W3C: HTML 5. <http://www.w3.org/TR/html5/>, Elérés ideje: 2014. november 9.
- [24] W3Schools: Browser Statistics. http://www.w3schools.com/browsers/browsers_stats.asp, Elérés ideje: 2014. november 15.
- [25] Dr. Kovács Zoltán: A hiperbolikus síkgeometria Poincaré-féle körmodellje. <http://zeus.nyf.hu/kovacs/poincare.pdf>, Elérés ideje: 2014. november 1.

A. A vizualizáció alapját képező táblázatok

A.1. táblázat. A böngészőpiac alakulását leíró táblázat

data	Chrome	IE	Mozilla	Safari	Opera
2008	4	46	44	3	2
2009	10	37	46	4	2
2010	22	28	44	4	2
2011	35	20	38	4	3
2012	47	15	31	4	2
2013	56	9	27	4	2
2014	60	10	23	4	2

A.2. táblázat. Az autómárkák és üzemanyag típusok kapcsolatát leíró táblázat

	data	diesel	gas
alfa-romeo	0	3	
audi	0	7	
bmw	0	8	
chevrolet	0	3	
dodge	0	9	
honda	0	13	
isuzu	0	4	
jaguar	0	3	
mazda	3	15	
mercedes-benz	4	4	
mercury	0	1	
mitsubishi	0	13	
nissan	2	17	
peugeot	5	6	
plymouth	0	7	
porsche	0	5	
renault	0	2	
saab	0	6	
subaru	0	7	
toyota	3	29	
volkswagen	4	8	
volvo	1	10	

A.3. táblázat. Az agy működésével kapcsolatos adatokat tartalmazó táblázat

data	interception	touch	pain	action	emotion	attention	memory	reward
interception	0	5	17	5	38	13	8	1
touch	5	0	1907	823	634	1229	829	104
pain	17	1907	0	9150	7023	5694	1958	335
action	5	823	9159	0	2435	6850	5074	1327
emotion	38	634	7023	2435	0	9294	9944	1481
attention	13	1229	5694	6810	9294	0	21850	1360
memory	8	820	1958	5074	9944	21850	0	1774
reward	1	104	335	1327	1481	1369	1774	0

A.4. táblázat. Arduino eszközökkel kapcsolatos felmérés eredményét tartalmazó táblázat

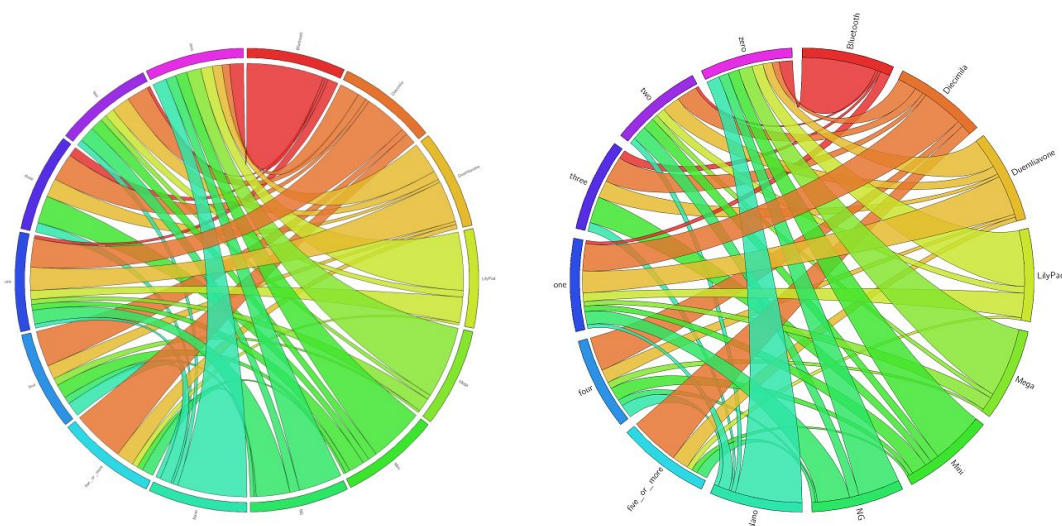
data	zero	one	two	three	four	five or more
Duemliavone	21	28	6	2	2	3
Diecimila	16	35	7	3	5	9
NG	22	17	5	0	1	2
Nano	30	5	2	1	2	0
Mega	31	6	2	0	1	0
Bluetooth	30	4	1	1	0	0
LilyPad	25	11	3	0	0	1
Mini	25	7	4	3	2	1

A.5. táblázat. Ip címek kapcsolatát leíró táblázat

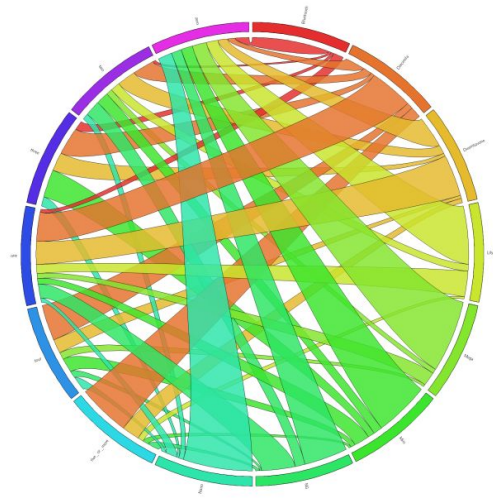
data	205.218.249.167	158.64.60.71	194.154.192.1	202.30.242.24	200.151.67.119	61.184.221.42	192.168.1.2	207.46.134.155	200.207.190.33
205.218.249.167	0	0	0	0	0	0	1	0	0
158.64.60.71	0	0	0	0	0	0	179	0	0
194.154.192.1	0	0	0	0	0	0	5	0	0
202.30.242.24	0	0	0	0	0	0	1	0	0
200.151.67.119	0	0	0	0	0	0	1	0	0
61.184.221.42	0	0	0	0	0	0	1	0	0
192.168.1.2	1	174	5	1	1	1	0	9	1
207.46.134.155	0	0	0	0	0	0	0	0	0
200.207.190.33	0	0	0	0	0	0	1	0	0

B. További összehasonlító ábrák

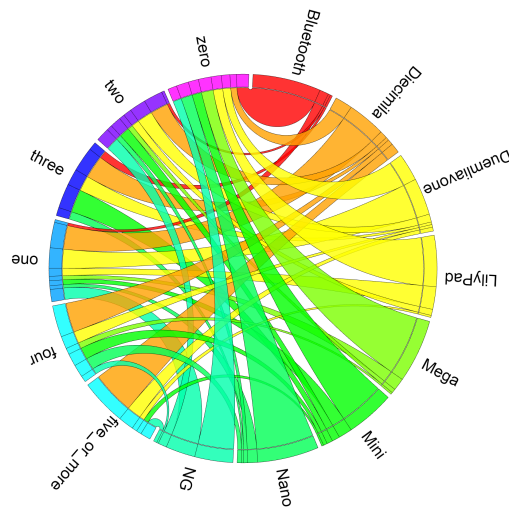
Arduino felmérés vizualizációja (lásd B.1. – B.3. ábra)



B.1. ábra. A Circos (bal oldalon) és a Circos Table Viewer (jobb oldalon) alapértelmezett kimenetei az A.5. táblázat adataira

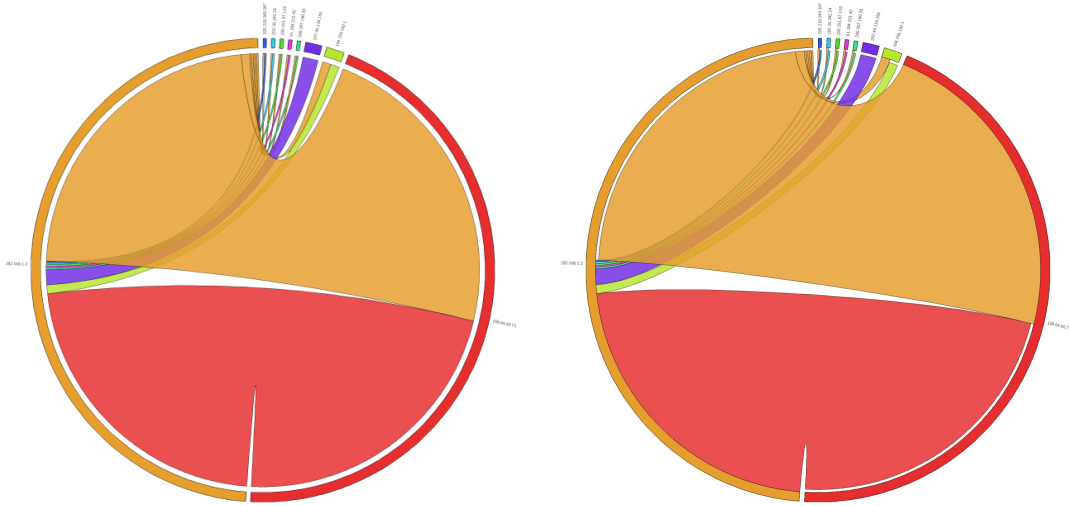


B.2. ábra. A Circos alkalmazással nyert kimenetek az A.5. táblázat alapján. A kép paraméterei: (brp: 0.025, br: 0.75, c: 1)

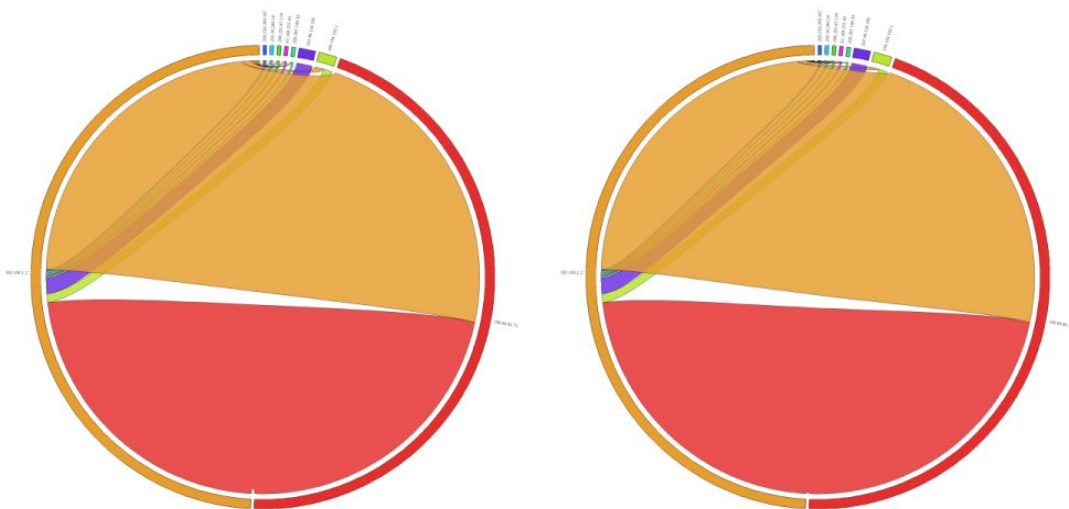


B.3. ábra. A saját szoftverünk (Poincaré Viewer) kimenete az A.5. táblázat adataira

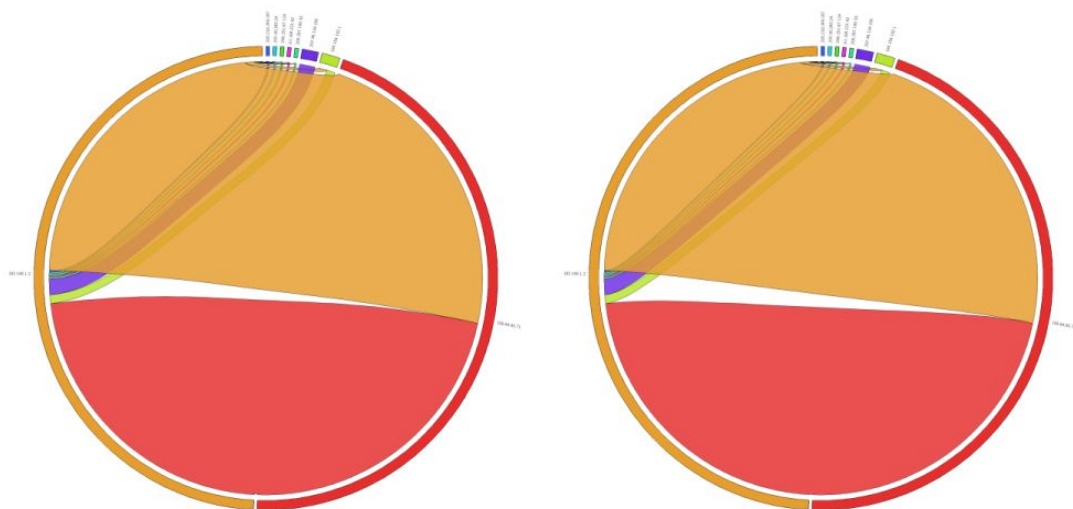
IP címek közötti kapcsolatot reprezentáló táblázat vizualizációja (lásd B.4. – B.7. ábra)



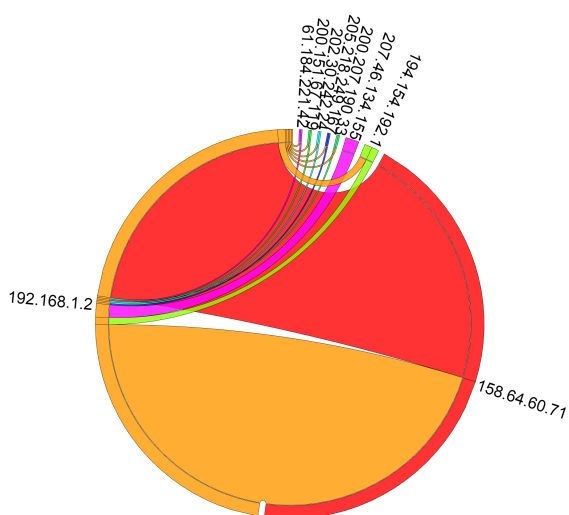
B.4. ábra. A Circos (bal oldalon) és a Circos Table Viewer (jobb oldalon) alapértelmezett kimenetei az A.4. táblázat adataira



B.5. ábra. A Circos alkalmazással nyert kimenetek az A.4. táblázat alapján. A jobb kép paraméterei (brp: 0.05, br: 0.2), bal kép paraméterei (brp: 0.025, br: 0.8)



B.6. ábra. A Circos alkalmazással nyert kimenetek az A.4. táblázat alapján. A jobb kép paramétere (brp: 0.025), bal kép paramétere (brp: 0.0001)



B.7. ábra. A saját szoftverünk (Poincaré Viewer) kimenete az A.4. táblázat adataira