SZAKDOLGOZAT

PrExt algoritmusok

Előry Zoltán

Témavezető: Hujter Mihály
egyetemi docens

BME Matematika Intézet,
Differenciálegyenletek Tanszék

BME
2010

Szakdolgozat-kiírás

Témavezető: Hujter Mihály, egyetemi docens
Téma: Gráfok színezéseinek kiterjesztései

A gráfok színezéseinek kiterjesztése az utolsó egy-másfél évtizedben az érdeklődés tárgyát képezte. Ennek egyik oka színezéskiterjesztések alkalmazhatósága gyakorlati feladatoknál, a másik az algoritmuselméletben betöltött szerepük. A diplomamunka készítőjének feladata lenne egyrészt áttekinteni a kutatási ág jelenlegi helyzetét, másrészt egy napjainkban is vizsgált részterületen egy-két speciális kérdést alaposan is meg kellene tárgyalnia. Megfelelő táblázatokat és színes ábrákat is várunk.

Irodalom:

http://www.zib.de/Publications/Reports/SC-97-31.pdf
http://www.math.bme.hu/ hujter/prext1.pdf
http://www.math.bme.hu/ hujter/prext2.pdf
http://www.math.bme.hu/ hujter/prext3.pdf

Tartalomjegyzék

1. Bevezetés

1.1. A dolgozat témája

A dolgozatban a mai matematika egyik leggyorsabban fejlődő és egyben egyik legtöbb alkalmazással bíró ágáról, a gráfelméletről írok. Ezen belül az úgynevezett gráfelőszínezés kiterjesztési kérdésekre térek ki. Ehhez az 1. fejezetben felidézek a témához kapcsolódó és a későbbiekben használt fogalmakat és tételeket. A 2. fejezetben ismertetem a $PrExt$ (Precoloring Extension) fogalmát és körüljárom az intervallumgráfokhoz köthető eredményeket, melyek a témában születtek. A 2.5 alfejezetben elemzem a függelékben csatolt, `Mathematica`-ban írt programomat, mely az $1 - PrExt$ feladatot oldja meg intervallumgráfokra. A 3. fejezetben egy minimális áttekintést nyújtok a merev körű gráfokról és kapcsolatáról az intervallumgráfokkal. A függelékben közlöm a fent említett forráskódot, illetve egy példával szemléltetem a működését.

1.2. Példa gyakorlati alkalmazásra

Példának egy, a való életben előforduló alkalmazást említek. Ez a probléma a MALÉV-nél merült fel. Legyen adott n feladatunk, jelölje ezeket rendre: F_1,F_2,...,F_n. Minden egyes F_i feladathoz rendeljünk egy (a_i, b_i) nyílt intervallumot, ahol $0 \leq a_i < b_i$. A F_i feladatot ezen időintervallumban kell végrehajtani. Emellett adott még $k \geq 2$ egyforma gépünk, legyenek ezek R_1,R_2,...,R_k. Az elvégzendő feladatoknak két típusa van: $\forall\ 0 \leq i \leq m$ esetén egy adott R_j gép már hozzá van rendelve az adott F_i feladathoz, illetve $\forall\ m+1 \leq i \leq n$ esetén az adott F_i feladathoz még hozzá kell rendelnünk egy megfelelő M_j gépet. Minden gép természetesen egyszerre csak egy feladat elvégzésére használható. Egy megfelelő hozzárendelés egy fízibilis ütemezést jelent, azaz az összes, k^{n-m} lehetőségből ki kell választanunk egy megvalósíthatót, avagy be kell látni, hogy ilyen nem létezik.

Ennél a konkrét példánál a gépek adott típusú repülőgépek, ahol k értéke körülbelül 5, és a teljes időintervallum egy adott hét, a_i és b_i pedig a hét adott órái. Az első F_1,F_2,...,F_m feladatok a repülőgépek karbantartási idejéhez rendelt feladatok. Általában egy adott repülőgépet egy negyedévben egyszer vagy kétszer kell karbantartani, ezért m értéke 5 és 10 közé tehető. Az időintervallumok, amikor is egy repülőgépen a szükséges ellenőrzéseket és javításokat végre kell hajtani, fixek és egy előre meghatározott terv alapján lettek lerögzítve. A maradék F_{m+1},F_{m+2},...,F_n feladat az adott negyedévben végrehaj-

tandó repülések időintervallumaihoz rendelt feladatok. Ezekben az időintervallumokban nem csak a repülés ideje, de az adott reptereken eltöltött idő is bele lett kalkulálva. Minden út a Budapest Airport-nál kezdődik, és ez a reptér a végállomás is. Természetesen minden egyes repülőgép adott időben csak egy utat tud lebonyolítani, ám szabad kezet kapunk abban, hogy ezt melyik repülőgép hajtsa végre, amennyiben az adott gép épp nincs karbantartás alatt. Általában adott típusú repülőgéppel egy negyedévben 50 utat kell végrehajtani.

Ez az ütemezési feladat igazából egy intervallumgráfokra kiírt PrExt feladat. Legyenek a gráf csúcsai az F_1,F_2,...,F_n feladatok. Egy adott csúcs megszínezése annyit tesz, mint egy adott gépet rendelni a csúcshoz tartozó feladathoz. Az előszínezett csúcsok a karbantartási feladatoknak felelnek meg, hiszen ezek tőlünk függetlenül már adottak. Két csúcs közt akkor és csak akkor megy él, ha ezekhez tartozó időintervallumok metszete nem üres halmaz. A gráf színezése tehát egy megfelelő hozzárendelést fog eredményezni a repülőgépek és feladatok között.

1.3. Gráfelméleti alapok

Mielőtt belemerülnénk a gráfelőszínezések kiterjesztési kérdésekbe, először felidéznék egynéhány definíciót és tételt, melyeket a továbbiakban használni fogunk.

1.3.1. Alapfogalmak

Definíció: Egy **gráf** egy rendezett pár, $G = (V, E)$, ahol V egy nem-üres halmaz, E pedig ebből a halmazból képezhető párok halmaza. Ekkor V elemeit **pontoknak** vagy **csúcsoknak** nevezzük, E elemeit pedig **éleknek** nevezzük.

Definíció: Két csúcs **szomszédos**, ha van közöttük él.

Definíció: **Hurokélnek** nevezünk egy olyan élet, melynek a két végpontja megegyezik. Ha két különböző hurokélnek a végpontjai azonosak, akkor a két élet **párhuzamos** vagy **többszörös élnek** nevezzük.

Definíció: Egy gráf **egyszerű**, ha nincs hurokéle és nincs többszörös éle.

Definíció: Egy csúcs **fokszámán** értjük a rá illeszkedő élek számát. Egy G gráf legkisebb fokszámú csúcsának a fokszáma a gráf **minimális fokszáma**, melyet $\delta(G)$-vel

jelölünk. Hasonlóan definiálhatjuk G **maximális fokszámot**, melyet $\Delta(G)$-vel jelölünk.

Definíció: A $G' = (V', E')$ gráf a $G = (V, E)$ gráf **részgráfja**, ha $V' \subseteq V$, $E' \subseteq E$ valamint egy pont és egy él pontosan akkor illeszkedik egymásra G'-ben, ha G-ben is illeszkedők. Ha E' pontosan azokból az E-beli élekből áll, amelyeknek mindkét végpontja V'-ben van, és E' az összes ilyen élet tartalmazza, akkor G' a G gráf V' által **feszített részgráfja**, más néven **indukált részgráfja**.

Definíció: Ha egy egyszerű gráf tetszőleges két pontja szomszédos, akkor **teljes gráfnak** nevezzük.

Definíció: G egy teljes részgráfját **klikknek** nevezzük. A G-ben található maximális méretű klikk méretét a **gráf klikkszámának** nevezzük és $\omega(G)$-vel jelöljük.

Definíció: $X \subseteq V(G)$ **független csúcshalmaz**, ha nincs benne két szomszédos csúcs. G legnagyobb független csúcshalmazának méretét $\alpha(G)$-vel jelöljük.

Definíció: Egy egyszerű G gráf **komplementere** az az egyszerű $\overline{G}$ gráf, amelynek a csúcshalmaza megegyezik G csúcshalmazával, és $\overline{G}$-ben pontosan azok a pontpárok vannak összekötve, melyek G-ben nincsenek.

Definíció: Egy olyan $(v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k)$ sorozatot, ahol a csúcsok különbözőek, és az e_i a v_{i-1}-et és v_i-t összekötő él, **útnak** nevezünk.

Definíció: Egy olyan $(v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k)$ sorozatot, ahol $e_1 \neq e_2$, $v_0 = v_k$ és különben a csúcsok mind különbözőek **körnek** nevezünk.

Definíció: A gráf egy **komponenséhez** azok a csúcsok tartoznak, melyek között van út.

Definíció: Azokat az éleket nevezzük **elvágó éleknek**, melyek elhagyásával a gráf komponenseinek száma nő.

Definíció: Egy gráf **összefüggő**, ha csak egy komponense van.

Definíció: Egy gráfot akkor nevezünk **irányítottnak**, ha megkülönböztetjük az éleinek a kezdőpontját és végpontját.

Definíció: Az összefüggő körmentes gráfokat **fának** nevezzük.

A hálózati folyam és a gráfszínezés definícióit és ismertebb tételeit, a szakdolgozatban betöltött fontos szerepe miatt, külön szakaszban közlöm.

1.3.2. Hálózati folyamok

Definíció: Legyen G egy irányított gráf. Rendeljünk minden e éléhez egy $c(e)$ nemnegatív valós számot, amit az él **kapacitásának** nevezünk. Jelöljünk ki továbbá két s, t pontot G-ben, melyeket **termelőnek\forrásnak** és **fogyasztónak\nyelőnek** hívunk. Ekkor a (G, s, t, c) négyest **hálózatnak** nevezzük.

Definíció: Legyen $f : E(G) \to \mathbb{R}^+ \cup \{0\}$ egy függvény, melyre teljesülnek:

- $f(e) \leq c(e) \quad \forall\, e \subseteq E(G)$
- $m(v) = \sum\{(f(e)|e \text{ végpontja } v)\} - \sum\{(f(e)|e \text{ kezdőpontja } v)\} = 0 \quad \forall\, v \subseteq V(G)\backslash\{s,t\}$

Ekkor f függvényt **folyamnak**, $m(t) = -m(s)$ értéket pedig a **folyam értékének** nevezzük és m_f-el jelöljük. Egy élet **telítettnek** hívunk egy folyamban, ha $f(e) = c(e)$, illetve **telítetlennek**, ha $f(e) < c(e)$.

Definíció: Legyen $s \in X \subseteq V(G)\backslash\{t\}$, így nyilvánvaló, hogy sem X, sem $V(G)\backslash X$ nem üres halmaz. Azoknak az éleknek a C halmazát, amelyeknek egyik végpontja X-beli, másik $V(G)\backslash X$-beli, a hálózati folyam egy **(s,t)-vágásának** nevezzük. A **vágás értéke**: $c(C)$.

Gyakran előforduló feladat, hogy adott hálózat esetén határozzuk meg a maximális folyam értéket, illetve mutassunk egy ilyen maximális értékű folyamot. Ezen feladatok megoldásához nekünk az alábbi tételekre lesz szükségünk:

Tétel[Ford-Fulkerson]. A maximális folyam értéke egyenlő a minimális vágás értékével.

Tétel[Edmonds-Karp]. A maximális folyam meghatározható $O(|V|^2 \cdot |E|)$ időben.

Tétel. Ha a kapacitások egész számok, akkor a maximális folyam értéke egész szám, és ez olyan f függvénnyel is megvalósítható, mely minden élen egész értéket vesz fel.

1.3.3. Gráfok színezése

Definíció: Egy G hurokmentes gráf k **színnel kiszínezhető**, avagy k**-színezhető**, hogyha minden csúcsot ki lehet színezni k szín felhasználásával úgy, hogy bármely két szomszédos csúcs színe különböző legyen. G **kromatikus száma** $\chi(G) = k$, ha G k színnel kiszínezhető, de $k-1$ színnel nem. Egy ilyen színezésnél az azonos színt kapott pontok halmazát **színosztálynak** nevezzük.

Nyilvánvalóan ha egy hurokél csatlakozna egy ponthoz, akkor azt a pontot nem lehetne kiszínezni. Másrészt a színezés szempontjából a többszörös élek nem számítanak, hiszen semmilyen színezést megváltoztató tulajdonságuk nincs, ezért elég a színezési kérdések esetében egyszerű gráfokkal foglalkozni.

Tétel. Ha van a gráfban egy klikk, akkor ennek semelyik két pontja nem lehet azonos színű, azaz minden G gráfra teljesül, hogy $\chi(G) \geq \omega(G)$.

Ez az alsó becslés korántsem éles minden gráfra, sőt, léteznek olyan gráfok, amikre ez a korlát egy nagyon rossz becslés. Ennek alátámasztásra idézzük fel az alábbi tételt:

Tétel(Mycielski konstrukciója). Minden $k \geq 2$ egész számra van olyan G_k gráf, hogy $\omega(G_k) = 2$ és $\chi(G_k) = k$.

A tétel bizonyítása során megmutatjuk, hogyha már van egy G gráfunk, melyre $\chi(G) = k$ és $\omega(G) = 2$, akkor ebből konstruálhatunk egy olyan gráfot, melyre már $\chi(G) = k+1$ és $\omega(G) = 2$. Ez alapján láthatjuk, hogy a klikkszám segítségével nem tudunk jó felső becslést adni a kromatikus számra. Helyette a maximális fokszám segítségével adható egy triviális felső becslés:

Tétel. Minden G gráfra teljesül, hogy $\chi(G) \leq \Delta + 1$.

Ha egy mohó színező algoritmust alkalmazzuk a gráfra, akkor az pont egy ilyen, legfeljebb $\Delta + 1$ színt felhasználó gráfszínezést fog eredményezni. Az alábbi tétel pontosítja ezt a becslést:

Tétel(Brooks). Ha G egyszerű, összefüggő gráf, nem teljes gráf, és nem egy páratlan hosszú kör, akkor $\chi(G) \leq \Delta$.

Ugyanazt elmondhatjuk most is, mint az alsó becslés esetén, ugyanis ez a becslés is lehet nagyon pontatlan. Vegyünk példának egy n pontú csillagot, azaz egy olyan gráfot, melynek egy középső pontjához csatlakozik $n-1$ él. Erre a gráfra teljesül, hogy $\Delta = n-1$, viszont $\chi(G) = 2$.

2. Előszínezések kiterjesztése

Ebben a szakaszban a gráfelőszínezések kiterjesztési kérdéseivel ismerkedünk meg behatóbban. Sok gyakorlati kérdés a kombinatorikus optimalizációban és a gráfelméletben lényegében visszavezethető egy gráfelőszínezés kiterjesztési kérdésre. Elég a bevezetőben ismertetett ütemezési feladatra gondolni. További példának még megemlíteném a VLSI (Very-Large-Scale-Integration) logikai árakmkört, melynek szerkezeti sajátosságai szintén gráfelméleti megfontolásokra vezethetőek vissza, azon belül is jelentős szerepet juttatva a gráfszínezési kérdéseknek. Ezeket a gráfelőszínezés kiterjesztési kérdéseket egységesen $PrExt$ kérdéseknek nevezzük, az angol *Precoloring Extension* megnevezésből rövidítve.

2.1. Előszínezés kiterjesztési feladat definiálása

Az eddigi jelöléseket használva: Legyen $G = (V, E)$ gráf, a gráf maximális klikkszáma $\omega(G)$ és a kromatikus száma $\chi(G)$. Legyen adott továbbá egy $W \subseteq V$ részhalmaz és egy $|V| \geq k \geq 2$ egész szám. Ekkor a PrExt az alábbi módon fogalmazható meg:

Feltéve, hogy adott $G[W]$ indukált részgráfon egy k-színezés, kiterjeszthető-e ez a színezés az egész G gráfra?

Sajnos ezek a $PrExt$ kérdések számos nehézségbe ütköznek. Vizsgáljuk meg az alábbi példát. Legyen $k \geq 2$ egy egész szám és legyen n k-nak egy adott többszöröse ($n \geq 2k$). Legyen $V = \{1, 2, \ldots, n\}$ és definiáljuk $G = (V, E)$ gráfot. Egy adott $uv \in E$ akkor és csak akkor, ha $u, v \in \{1, 2, \ldots, n\}$ és $0 < |u - w| < k$. Ekkor ez a G gráf perfekt és $\chi(G) = \omega(G) = k$. A perfektséget könnyedén beláthatjuk, ha adunk egy jó intervallum reprezentációt, azaz belátjuk, hogy G intervallumgráf. Egy ilyen megadása lehet G-nek, ha $\forall\ i \in \{1, 2, \ldots, n\}$ legyen $a_i = i$ és $b_i = i + k$. Azaz ez a gráf k-színezhető. Azonban, ha megszínezzük az első és utolsó csúcsot azonos színnel, akkor ez az előszínezés már nem terjeszthető ki egy helyes k-színezésre. Tehát a $PrExt$ valóban legalább olyan nehéz feladat, mint a gráfszínezés.

Definiáljuk még a $d - PrExt$-et. Ez a PrExt kérdéskörnek egy szűkebb része. Legyen k színkorlát lerögzítve és legyen d egy nemnegatív egész szám. Ekkor a $d - PrExt$ csak azokra a k színnel előszínezett gráfokra vizsgálja a $PrExt$ kérdést, amelyeknél az előszínezés során minden színt legfeljebb d alkalommal használtunk fel. Így például a $0 - PrExt$ a szokásos gráfszínezési kérdés.

2.2. Intervallumgráfok

Definíció: Legyen A egy halmaz és S egy A nem üres részhalmazaiból álló halmaz egy részhalmaza. Ekkor S **metszetgráfja** úgy áll elő, ha a gráf csúcsai elemei S-nek, és két csúcs akkor és csak akkor szomszédos, ha a nekik megfelelő részhalmazok metszete nem üres. Ha a halmazok valós számokon vett zárt intervallumok, akkor a gráfot **intervallumgráfnak** nevezzük.

Tétel. Minden intervallumgráf perfekt.

Bizonyítás: Legyen G egy intervallumgráf, $V(G) = \{v_1, v_2, \ldots, v_n\}$ a csúcsok halmaza, valamint $C = \{c_1, c_2, \ldots\}$ a felhasználandó színek halmaza. Mivel minden intervallumgráfnak minden feszített részgráfja is intervallumgráf, ezért elég azt megmutatni, hogy $\chi(G) = \omega(G)$. Mivel G intervallumgráf, ezért létezik n darab olyan $I_i = [a_i, b_i]$, $1 \leq i \leq n$ zárt intervallum, hogy v_i akkor és csak akkor szomszédos v_j-vel, ha $I_i \cap I_j \neq \emptyset$. Feltehetjük, hogy az intervallumok és vele együtt G csúcsai is úgy lettek felcímkézve, hogy $a_1 \leq a_2 \leq \ldots \leq a_n$. Most megadunk egy színezést G gráfra.

Legyen v_1 a c_1 színnel megszínezve. Amennyiben v_2 nem szomszédos v_1-gyel (azaz $I_1 \cap I_2 = \emptyset$), úgy kapja meg ő is a c_1 színt, ellenkező esetben c_2-t. Induktívan tovább haladva, tegyük fel, hogy már megszíneztük $v_1, v_2, \ldots, v_r$ csúcsokat ($1 \leq r < n$). Ekkor v_{r+1} kapja meg a legkisebb indexű c_i színt, melyet egyik $\{v_1, v_2, \ldots, v_r\}$-beli szomszédja sem kapott még meg. Amennyiben v_{r+1} egyik előző halmazbeli csúccsal sem szomszédos, úgy c_1 színnel színezzük meg. Így kaptunk egy helyes k-színezést G gráfra valamely pozitív egész k-ra, azaz $\chi(G) \leq k$. Ha $k = 1$, akkor a gráfunkban nincs él, azaz triviálisan $\chi(G) = \omega(G) = 1$. Így a továbbiakban nyugodtan feltehetjük, hogy $k \geq 2$.

Most tegyük fel, hogy a v_t csúcs a c_k színt kapta. Ez azt jelenti, hogy nem volt lehetséges, hogy $c_1, c_2, \ldots, c_{k-1}$ színek bármelyikét megkaphassa, vagyis $I_t = (a_t, b_t)$ intervallumnak $k-1$ intervallummal volt nem üres metszete. Tehát létezik $I_{j_1}, I_{j_2}, \ldots, I_{j_{k-1}}$ intervallum, melyekre $1 \leq j_1 < j_2 < \ldots < j_{k-1} < t$ és $I_{j_i} \cap I_t \neq \emptyset$ $\forall\ 1 \leq i \leq k-1$. Ebből már láthatjuk, hogy ez csak akkor lehetséges, ha $a_t \in I_{j_1} \cap I_{j_2} \cap \ldots \cap I_{j_{k-1}}$. Vegyük a $\{v_{j_1}, v_{j_2}, \ldots, v_{j_{k-1}}, v_t\}$ által indukált részgráfot. Mivel a_t minden hozzátartozó intervallumnak eleme, így bármely kettő csúcs szomszédos,

azaz ez a gráf csak a K_k lehet. Vagyis $\chi(G) \leq k \leq \omega(G)$. De mivel tudjuk azt is, hogy a $\chi(G) \geq \omega(G)$, ezért megkaptuk, hogy $\chi(G) = \omega(G)$.

■

2.3. $1 - PrExt$ intervallumgráfokra

A fejezet célja, az alábbi tétel bizonyítása:

Tétel. Az $1-PrExt$ feladat intervallumgráfok esetében megoldható polinomiális időben.

Bizonyítás: Legyen $G = (V, E)$, $V = \{1, 2, \ldots, n\}$ egy intervallumgráf, adott (a'_i, b'_i) $i = 1, 2, \ldots, n$ intervallumokra. Ezeket az intervallumokat megadhatjuk egy szebb alakban is. Rendezzük az intervallumokat b'_i szerinti növekvő sorrendbe. Ekkor az intervallumok határait megváltoztathatjuk úgy, hogy $\forall\ 1 \leq i \leq n$-re $0 \leq a_i < b_i = i$ teljesüljön, és továbbra is G intervallum reprezentációja legyen. Egy ilyen szép alakra hozás megvalósítható polinomiális időn belül. A továbbiakban használjuk G ezen intervallum reprezentációját.

Tehát legyen $G = (\{1, 2, \ldots, n\}, E)$ egy intervallumgráf, melyet generáljunk az $a_1 = 0, a_2, \ldots, a_n$ sorozattal, és legyen $W \subseteq \{1, 2, \ldots, n\}$ egy előszínezett részhalmaza a csúcsoknak. Legyen k a G gráf $1-PrExt$ színkorlátja. Ha $k < \omega(G)$, akkor az intervallumgráfok perfektsége miatt az $1 - PrExt$-re egy triviális nem a válasz. $\omega(G)$ értékét lineáris időben meg tudjuk keresni. Így a továbbiakban tegyük fel, hogy $k \geq \omega(G)$.

Konstruáljunk egy új, körmentes, irányított $D(G, W, k)$ gráfot. Legyen ebben $V(D) = \{0, 1, 2, \ldots, n, n+1\}$. Az irányított éleket 3 csoportba sorolhatjuk: eredeti, kisegítő és módosított. Az eredeti élek legyenek $\overrightarrow{a_i i}$ alakúak, ahol legyen $\forall\ i \in \{1, 2, \ldots, n\} \backslash W$. Ezek az élek megfeleltethetőek az eredeti G gráf előszínezetlen csúcsaival. A kisegítő élek legyenek $\overrightarrow{(i-1)i}$ alakúak $i = 1, 2, \ldots, n, n+1$-re. Végül legyenek a módosított élek $\overrightarrow{a_i(n+1)}$ $\forall\ i \in W$.

Rendeljünk nemnegatív egész kapacitásértékeket minden élhez. Az eredeti és módosított élek mind 1-es kapacitásértéket kapnak. A kisegítő élekhez az alábbi formulával adjuk meg a kapacitást:

$\forall\ i \in \{1, 2, \ldots, n\}\ \ \overrightarrow{(i-1)i}\ \rightarrow\ (k - |\{a_j : a_j < i \subseteq j|)$ és $(n, n+1)\ \rightarrow\ (k - |W|)$.

Megmutatjuk, hogy akkor és csak akkor létezik 0 és $n+1$ közt egy k-folyam D-ben, ha a G-beli előszínezés kiterjeszthető egy helyes k-színezésre. Ehhez tegyük fel először, hogy az előszínezés valóban kiterjeszthető egy $f : V \rightarrow 1, 2, \ldots, k$ színezésre. Ha az i színt még nem használtuk fel az előszínezés során, akkor legyen $V_i = f^{-1}(i)$. Máskülönben ha egy (a_j, j) intervallum már megkapta az i színt az előszínezés során, akkor legyen V_i azon $f^{-1}(i)$-beli csúcsok, melyekre az intervallumok jobb végpontja legfeljebb j. Láthatjuk, hogy V_i halmazok mind egy folyamot adnak D-ben, így ezek uniója megad egy k-folyamot D-ben.

A másik irány bizonyításához tegyük fel, hogy van egy F k-folyamunk D-ben. Mivel minden élen a kapacitás egész szám, ezért ez a folyam legyen olyan, hogy minden élhez egész számot rendel. Legyen D' egy olyan irányott gráf, melyet D-ből állítunk elő úgy, hogy minden D-beli, c kapacitású kisegítő él helyére c darab, független $\overrightarrow{(i-1), i}$ élet helyettesítünk. Mivel D-ben létezik k folyam, ezért D'-ben létezik k darab élfüggetlen út. Legyenek ezek $P_1, P_2, \ldots, P_k$. Minden P_i út legfeljebb egy módosított élet tartalmazhat és minden módosított él hozzátartozik valamelyik P_i úthoz, hiszen $\overrightarrow{n, n+1}$ élhez $k - |W|$ kapacitást rendeltünk.

Térjünk vissza G gráfhoz és adjunk hozzá a D-beli kapacitásaiknak megfelelő számú egységnyi $(i-1, i)$ intervallumot. Legyen ez az új konstrukció S. Ekkor minden D'-beli él, az $(n, n+1)$ élen kívül megfeleltethető egy S-beli intervallumnak. Azaz P_i utak is megfeleltethetőek egymást követő S-beli intervallumok sorozatának, melyek 0-ból indulnak. Vegyük ezen utak konvex burkát, azaz minden P_i út mellé vegyük egy minimális H_i intervallumot, melyre $(a_j, b_j) \in P_i \Rightarrow (a_j, b_j) \in H_i$. Most töröljük ki mindegyik P_i-hez tartozó intervallumot S-ből, és helyette vegyük be az összes H_i intervallumot a rendszerbe. Ezt az új konstrukciót nevezzük S'-nek.

Végül vegyük G' gráfot, mely az S' intervallumainak metszetgráfjaként állítunk elő. Ez is intervallumgráf, azaz a perfektség miatt $\chi(G') = \omega(G')$, $\omega(G') = k$ és ebben a gráfban nincs előszínezett csúcs. Tehát ezt az intervallumgráfot egy mohó színező algoritmussal megtudjuk színezni k színnel polinomiális időben, és mivel H_i intervallumok metszőek (mindegyik 0-ból útból lett generálva), ezért különböző színeket kapnak a színezés során. Feltehetjük, hogy minden H_i intervallumot i színnel színeztük meg. Ebből vissza tudunk vezetni egy jó előszínezés kiterjesztést

G-re. Legyen $\forall\, v \in V\ f(v) = i$ akkor és csak akkor, ha P_i út tartalmazza a v csúcshoz tartozó intervallumot, vagy a v-t reprezentáló intervallum S'-ben nem metsző H_i-vel és G' színezése során az i színt kapta. Az így definiált f színezés egy jó kiterjesztése lesz a G-beli előszínezésnek. Mivel F folyamot megkereshetjük polinomiális időben (vagy bizonyíthatjuk, hogy nincs ilyen), ezért a tételt bizonyítottuk.

■

2.4. $2 - PrExt$ intervallumgráfokra

Miután beláttuk, hogy az $1 - PrExt$ megoldaható polinomiális időben, érdemes megvizsgálni, vajon fennáll-e ez az állítás $2 - PrExt$ esetében is, illetve általánosabban: $k \geq 2$ esetén megoldható-e a $k - PrExt$ az intervallumgráfokra polinomiális időben. A válasz az, hogy ezekben az esetekben már NP-teljes feladatokkal van dolgunk. Mielőtt ezt belátnánk, nézzük meg a körívgráfok fogalmát és két rájuk vonatkozó tételt, melyeket felhasználunk a bizonyításban.

Definíció: A **körívgráf** egy körön vett, zárt körívekből képzett metszetgráf.

Minden intervallumgráf körívgráf is egyben, hisz elég felfűzni az intervallumokat egy kellő méretű körre, így minden intervallum reprezentációhoz rendelhetünk egy neki megfelelő körív reprezentációt. Fordítva már nem igaz, elég ellenpéldaként egy 4 hosszú körre gondolni, melynek intervallum reprezentációja nem létezik, ellenben körív reprezentációja triviális.

Tétel. Egy körívgráf felismerhető $O(|V|^3)$ időben, és egy neki megfeleltethető körív reprezentáció megkereshető polinomiális időben.

Tétel. Egy nem korlátos k pozitív egészre, a " $\chi(G) \leq k$? " gráfszínezési kérdés NP-teljes.

Ezen ismeretekkel felvértezve, már beláthatjuk az alábbi tételt:

Tétel. A $2 - PrExt$ NP-teljes intervallum gráfok esetén.

Bizonyítás: Vegyünk egy tetszőleges $G = (V, E)$ körívgráfot és egy tetszőleges k pozitív egész számot. Tegyük fel, hogy $V = \{1, 2, \ldots, n\}$, és minden körív egy útnak felel

meg a $C = (\{1, 2, \ldots, m\}, \{12, 23, \ldots, (m-1)m, m1\})$ gráfban, ahol m egy kellően nagy, pozitív, egész szám. Tegyük fel továbbá, hogy minden körív legalább 2 és legfeljebb $m-1$ C-beli csúcsot tartalmaz. Ekkor azok a körívek, melyek az $m1$ C-beli élet tartalmazzák egy klikknek felelnek meg G-ben. Legyen ezen csúcsok száma t, melyre $t \leq \omega(G)$. Ekkor $\forall\ 1 \leq i \leq t$ esetén $\exists P_i$ C-beli út, mely tartalmazza az $m1$ élet.

Konstruáljunk egy új G' gráfot az alábbi módon. Minden P_i utat bontsunk két rövidebb, P_i' és P_i'' útra az $m1$ él elvételével. Így kaptunk egy intervallum reprezentációt $n+t$ intervallummal, ahol ezek az intervallumok $C\backslash\{m1\}$-beli utak. Egy ilyen G' gráf lineáris időben megépíthető G-ből. Most vegyünk egy előszínezést G'-n: $\forall\ 1 \leq i \leq t$ esetén rendeljük P_i' és P_i'' utakhoz az i színt, G' fennmaradó csúcsai pedig legyenek színezetlnek. Ezzel generáltunk egy $2 - PrExt$ kérdést egy nem korlátos k pozitív egészre. Ez az előszínezés G'-n akkor és csak akkor terjeszthető ki egy helyes k-színezésre, ha $\chi(G) \leq k$. De erről már tudjuk, hogy NP-teljes.

■

2.5. $1 - PrExt$ algoritmus implementálása `Mathematica`-ban

A szakdolgozat egyik fontos célkitűzése volt egy hatékony program megírása, mellyel eldönthetjük egy adott intervallumgráfról, melynek néhány csúcsát előre megszíneztünk minden szín legfeljebb egyszeri felhasználásával, hogy az előszínezése kiterjeszthető-e egy helyes színezéssé. Eme program megírására a `Mathematica` széles eszköztárát használtam fel. A választásom azért esett erre a programnyelvre, mert már rendelkezeik beépített, gráfok generálását, vizsgálását és kirajzoltatását megkönnyítő programcsomaggal. Sajnos az ezek használatát leíró részek néhol hiányosak, illetve nem egészen a kívánt eredménnyel valósítják meg a szükséges parancsokat, így a program megalkotása során számos helyen felbukkant a probléma, hogy a bemenetet át kellett alakítani a `Mathematica` által is elismert alakra, illetve a kapott eredményt is vissza kellett alakítani általunk használható formára. Ezekre a forráskód elemzése során részletesebben kitérek. A forráskódot magát teljes egészében közlöm a függelék első fejezéteben. Maga a program a 2.3-as szakaszban vázolt algoritmus elveit követi.

Az *első programrészben* az előzetes beállítások vannak. Ennek keretében elsőnek betöltöm a `Combinatorica` és `Graphutilities` csomagokat, melyek gráfokra alkalmazha-

tó parancsokat tartalmaznak, majd megadhatjuk intervallumokkal az intervallumgráfot, illetve beállíthatjuk, hogy ha nem adunk meg magunk egy ilyen gráfot, akkor a program milyen paraméterekkel generáljon egyet véletlenszerűen. A rész végén még beállíthatjuk, hogy a program lefutása után mely gráfokat és részeredményeket irassa ki.

A *második programrészben* generálunk egy random intervallumgráfot, amennyiben az előbeállítások során nem adtunk meg egyet. Az előbeállításban megadott csúcsszámnak megfelelő mennyiségű, 0 és `Intmax` közötti intervallumot állít elő.

A *harmadik programrészben* átalakítjuk ezeket az intervallumokat egy szebb alakra, mely sokkal könnyebben használható a későbbiekben. Az átalakítás itt is a 2.3-as szakaszban vázolt bizonyításbeli alakra történik. Ehhez előbb a `Sort` paranccsal rendezzük az intervallumokat a második paraméterük szerint, majd egy `For` ciklus keretében a megfelelő értékekre változtatjuk az intervallumhatárokat, végül ezt a `RendezettIntervallumok` változóban tároljuk.

A *negyedik programrészben* megadhatjuk, hogy mely csúcsok legyenek előszínezve a rendezett intervallumok közt, vagy amennyiben nem adunk meg, úgy generálunk egy véletlenszerű előszínezést. A `Drop` parancs használatával biztosítjuk, hogy egy csúcs legfeljebb csak egyszer szerepeljen a listában.

Az *ötödik programrészben* két `For` ciklussal legeneráljuk az intervallumgráf szomszédossági mátrixát.

A *hatodik programrészben* megkeressük a gráfban levő legnagyobb klikket. Erre a beépített `MaximumClique` parancsot használjuk, ahol is a gráfot a `FromAdjacencyMatrix` paranccsal a szomszédossági mátrixával adjuk meg. Amennyiben találunk egy, a megendegett színszámnál nagyobb klikket, akkor a program ugrik a kiirató részhez, és ismerteti ezen klikk csúcsait. Ebben az esetben a program többi része nem fordul le.

A *hetedik programrészben* legeneráljuk a `D` hálózathoz tartozó irányított éleket. Az `EredetiÉlek` pont az előszínezetlen élek, a `Drop` paranccsal az előszínezett élek eltávolításával állítjuk elő őket. A `KisegítőÉleket` egy egyszerű `Table` paranccsal megkapjuk. Végül a `MódosítottÉleket` az előszínezett intervallumokból nyerjük, a jobboldali intervallumhatár átállításával, majd sorbarendezéssel.

A *nyolcadik programrészben* a kapacitásokat számoltatom ki a különböző élekhez. Az `EredetiÉlekhez` és a `MódosítottÉlekhez` az 1 értéket rendelem, míg a `KisegítőÉlekre` a megadott képletet használom, az utolsó `KisegítőÉlet` pedig külön, a megfelelő értékre állítom.

A *kilencedik programrészben* az előbbi két rész összekapcsolásával megalkothatjuk a `D` hálózatot. Az `Élgráf` változó első felébe az irányított éleket helyezem az `Union` paranccsal, mely egyben sorba is rendezi őket. A változó második felébe a `capacitymatrixból` kiolvasott megfelelő kapacitásértékek kerülnek. Ennél a résznél jelentkeznek az első apróbb problémák, melyek a Matematica gráfkezelő részénél jelentkeznek. Bár a 2.3-as fejezetben az `D` hálózat 0-tól `(Csúcsszám+1)`-ig terjedt. Azonban a Matematica `Combinatorica` csomagja nem hajlandó kezelni 0 csúcsot, így az `Élgráfban` minden intervallumhatárt eggyel megnöveltem, és a későbbiekben is ennek megfelelően kell használni az intervallumokat.

A *tizedik programrészben* az `Élgráfot Matematica` által kezelhető, élkapacitásokkal rendelkező gráffá alakítom, majd kerestetek benne egy maximális folyamértéket, és egy ilyen értéket megvalósító folyamot a `NetworkFlow` paranccsal. Amennyiben ez a folyamérték kisebb, mint a felhasználható színek száma, úgy az előszínezés kiterjesztése nem megvalósítható. Ekkor a program ugrik a kiirató részhez, és közli, hogy nem talált elég nagy folyamot. A program további részei ekkor nem fordulnak le.

A *tizenegyedik programrészben* a talált folyamot alakítjuk át diszjunkt utakra, melyeket könnyebben kezelhetünk. Az elv maga egyszerű: `P` változóba legyen egy maximális folyam hosszúságú változó, amelynek minden egyes részébe egy neki megfelelő út intervallumait fogom betölteni. Amíg lehetséges, minden `P`-beli részhez rendelek egy `MódosítottÉlet`, a maradékhoz pedig a fennmaradó, `(Csúcsszám+2)` végű intervallumokból rendelek egyet-egyet. Ezután egy adott részhez addig fűzök hozzá balról megfelelő intervallumokat, míg el nem jutok a forrásig, közben pedig a felhasznált intervallumokat természetesen törlöm. Ezzel a módszerrel sikersen legenerálom az adott `P` utakat.

A *tizenkettedik programrészben* ebből a `P` utakból `Pvesszők` utakat állítok elő, hogy az utolsó útbeli intervallumokat kicserélem az eredeti, még nem módosított intervallumokra. Ehhez először kikeresem, hogy mik voltak az előszínezett intervallumok, majd törlöm a `P` listák utolsó elemét, és a megfelelő intervallumot a helyére illesztem.

A *tizenharmadik programrészben* veszem a `Pvesszők` utak konvex burkait. Ezek olyan intervallumok lesznek, melyeknek a baloldali intervallumhatára a forrás, a jobboldali intervallumhatára pedig az utolsó `Pvesszők`-beli intervallum jobboldali intervallumhatára.

A *tizennegyedik programrészben* legenerálom a `Gvesszők` gráfot. Itt javítok ki egy apróbb csalást, melyet még a program elején ejtettem. Ugyanis amikor a kapacitásmátrixot megadtuk, nem vettük figyelembe, hogy létezhetnek olyan élek is, melyek alapból `(i,i+1)` alakúak. Eddig ez az egyszerűsítés nem okoztott gondot, azonban az új

segédgráf definiálásánál előfordul, hogy az ilyen éleket túl sokszor, illetve túl kevésszer adjuk hozzá a gráfhoz. Ennek elkerülésére definiálom a `Javítóélek`-et, mely megszámolja, hogy pontosan hányszor történt ilyen mulasztás, és megfelelően korrigálja az élek számát. A `Gvesszős` felépítése ezen kívűl nem ütközik nagyobb problémába, beleveszem a `KisegítőÉleket`, elvégzem a szükséges javítást, majd kitörlöm a `Pvesszős` intervallumait és végül beleveszem a `H` intervallumokat.

A *tizenötödik programrészben* megkonstruálom a `Gvesszős` szomszédossági mátrixát, ugyanúgy, mint tettem ezt a `G` esetén az ötödik programrészben.

A *tizenhatodik programrészben* veszem a `Gvesszős` egy megfelelő színezését. Ezt az intervallumgráfok perfektségének bizonyításánál alkalmazott módszerrel teszem. Sorban minden intervallumhoz törlöm a nem lehetséges színeket, melyeket már egy korábbi szomszédja megkapott, majd a fennmaradó színek közül hozzárendelem a legkisebbet.

A *tizenhetedik programrészben* végre megvalósítom az eredeti intervallumgráf előszínezésének kiterjesztését. Minden intervallum vagy a neki megfelelő `H` színét kapja, vagy ha nincs benne egyik `H`-t generáló útban sem, akkor a `Gvesszőben` megkapott színt rendeljük hozzá.

A *tizennyolcadik programrészben* megkonstruálom a színosztályokat, majd mindegyikhez egy `Matematica` által felismert színt rendelek. Ez a színezett `G` kirajzoltatásához kell.

A *tizenkilencedik programrészben* kiiratjuk az előbeállítás során megjelölt paramétereket, illetve amennyiben nem volt lehetséges a színezés, egy rövid hibaüzenettel jelzem ennek okát.

3. Merev körű gráfok azonosítása

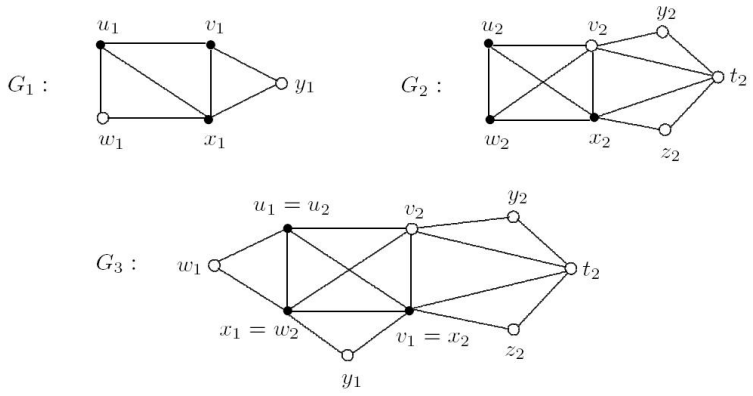
Ebben a fejezetben a merev körű gráfokat és azok egy néhány tulajdonságát mutatom be. A végső cél, egy olyan algoritmus bemutatása, mely polinomiális időn belül eldönti egy gráfról, hogy az merev körű avagy sem.

3.1. Merev körű gráfok tulajdonságai

Definíció: A G gráfot **merev körű gráfnak** nevezzük, ha minden, legalább 4 hosszú körében van húr, azaz egy olyan él, mely a kör két nem szomszédos csúcsát köti össze.

Minden teljes gráf merev körű is egyben. Ellenben a $K_{s,t}$ teljes páros gráfok nem lesznek ilyen tulajdonságúak, amennyiben $s, t \geq 2$. Vegyük tetszőleges $u_1, u_2, v_1, v_2 \in V(G)$ csúcsokat, melyek közül a párosítás során u_1, u_2 az egyik, míg v_1, v_2 a másik halmazbeli csúcsok. Ekkor ők négyen kifeszítenek egy 4 hosszú, húrmentes kört. Ezek alapján persze az is nyilvánvaló, hogy egyik gráf sem lehet merev körű, amennyiben tartalmaz feszített részgráfként egy háromnál hosszabb kört.

Most vizsgáljuk meg a merev körű gráfok néhány kellemes tulajdonságát. Tegyük fel, hogy G_1 és G_2 gráfok tartalmaznak részgráfként H_1 és H_2, azonos méretű klikkeket. Legyen G_3 az gráf, melyet G_1 és G_2 egyesítésével kapunk oly módon, hogy azonosítjuk H_1 csúcsait H_2 csúcsaival.



Vegyük észre, ha az új G_3 gráfban találunk egy legalább 4 hosszú húrmentes C kört, az csak úgy lehetséges, ha G_1 vagy G_2 gráf már tartalmazta ezt a C kört. Könnyen

belátható, hogy ha G_1 és G_2 merev körű gráfok, akkor G_3 is az, illetve ha G_3 merev körű, akkor G_1 és G_2 is az lesz.

Tétel. Legyen G egy olyan gráf, melyet két azonos méretű, G_1 és G_2 gráfbeli klikkek azonosításával állítunk elő. Ekkor G merev körű gráf akkor és csak akkor, ha G_1 és G_2 merev körű gráfok.

Bizonyítás: Azt már beláttuk, hogy ha G_1 és G_2 merev körű gráfok, melyek tartalmaznak azonos méretű H_1 és H_2 klikkeket, akkor ezen klikkek azonosításával előálló G gráf is merev körű. Másfelől tegyük fel, hogy G_1 nem merev körű, tartalmaz egy legalább 4 hosszú, húrmentes kört. Akkor az azonosítás után G ugyanúgy tartalmazni fogja ezt a kört, így ő sem lehet merev körű.

■

Ezen tétel felhasználásával beláthatjuk azt is, hogy egy gráf merev körűségének nem csak elégséges feltétele, hogy előálljon a fenti alakban, hanem szükséges is:

Tétel. Egy G gráf merev körű akkor és csak akkor, ha G előáll két merev körű gráfból, bennük található azonos méretű klikkek azonosításaként.

Bizonyítás: Az eddigi megfigyelésinkből következik, hogy elég belátni, hogy minden G merev körű előáll két merev körű gráfból, bennük található azonos méretű klikkek azonosításaként. Ha $G = K_n$ teljes gráf, akkor G merev körű, és ekkor előáll $G_1 = K_n$ és $G_2 = K_n$ azonosításaként, így a tétel igaz. Ezért a továbbiakban feltehetjük, hogy G egy összefüggő, nem-teljes merev körű gráf.

Vegyünk egy S minimális vágást G-n. Legyen V_1 az egyik, $G\backslash S$-beli komponens csúcshalmaza, és legyen $V_2 = V(G)\backslash(V_1 \cup S)$. Ezekből képezzük $G_1 = G[V_1 \cup S]$ és $G_2 = G[V_2 \cup S]$ gráfokat. Láthatjuk, hogy G előállítható, ha a G_1 és G_2 gráfokban azonosítjuk S csúcsait. Ha meg tudjuk mutatni, hogy S csúcsai egy teljes gráfot feszítenek ki, akkor kész a bizonyítás. Ha $|S| = 1$, akkor az állítás igaz, ezért tegyük fel, hogy $|S| \geq 2$.

Minden $v \in S$ esetén igaz, hogy v-nek létezik minden, vágás után keletkezett komponensben szomszédja, hiszen ha nem így lenne,akkor v elhagyásával egy kisebb vágás keletkezne, de feltettük, hogy S minimálás vágás. Legyen $u, w \in S$. Az

előbbi szerint ekkor léteznek $u-w$ utak G_1-ben. Az összes ilyen útból vegyünk egy minimális hosszút. Legyen ez $P = (u, x_1, x_2, \ldots, x_s, w)$. Hasonlóan vehetünk egy minimális hosszú $u-w$ utat G_2-ben is, legyen ez $P' = (u, y_1, y_2, \ldots, y_t, w)$. Ekkor

$$C = (u, x_1, x_2, \ldots, x_s, w, y_t, y_{t-1}, \ldots, y_1, u)$$

egy legalább 4 hosszú kör G-ben. G azonban merev körű, így biztosan létezik ebben a C körben húr. Ez a húr nem mehet egy x_i és y_j csúcsok közt, hisz akkor S nem lenne vágás. Nem mehet azonban két $P\backslash\{u,w\}$-beli csúcs közt sem, hisz akkor P nem lenne minimális út. $P'\backslash\{u,w\}$-beli csúcsokra hasonlóan. Azaz ez a húr csak u és w csúcsokra illeszkedhet. Ezzel beláttuk, hogy tetszőleges $u, w \in S$ csúcsok esetén $uw \in E(G)$, azaz $G[S]$ teljes gráf. Az előző tétel biztosítja, hogy G_1 és G_2 szintén merev körű gráfok.

■

Az fenti előállíthatóság következményeként beláthatjuk a következő fontos tételeket:

Tétel. Minden merev körű gráf perfekt.

Bizonyítás: Mivel merev körű gráfnak minden feszített részgráfja is merev körű, elég belátni, hogy ha G merev körű gráf, akkor $\chi(G) = \omega(G)$. Ezt a gráf n csúcsszámára végzett indukcióval bizonyítjuk. Ha $n = 1$, akkor $G = K_1$ és $\chi(G) = \omega(G) = 1$. Tegyük fel, hogy minden n-nél kevesebb csúcsú H merev körű gráfra teljesül, hogy $\chi(G) = \omega(G)$.

Legyen G egy n csúcsú merev körű gráf. Ha G teljes, akkor $\chi(G) = \omega(G) = n$. Tegyük fel, hogy G nem teljes. Az előző tétel szerint G előáll G_1 és G_2 merev körű gráfokból, bennük található azonos méretű klikkek azonosításaként. Így teljesül:

$$\chi(G) \leq \max\{\chi(G_1), \chi(G_2)\} = k.$$

Az indukciós feltevés szerint $\chi(G_1) = \omega(G_2)$ és $\chi(G_2) = \omega(G_2)$, azaz $\chi(G) \leq \max\{\omega(G_1), \omega(G_2)\} = k$. Másrészt legyen $S = V_1 \cap V_2$. Az előző tétel bizonyításában beláttuk, hogy $G[S]$ teljes , és $V_1\backslash S$ és $V_2\backslash$ közt nincs nem fut él. Ezek szerint:

$$\omega(G) = \max\{\omega(G_1), \omega(G_2)\} = k.$$

Mivel $\chi(G) \geq \omega(G)$, ezért $\chi(G) = k = \omega(G)$ is teljesül.

■

Definíció: Egy gráfbeli csúcsot **szimpliciálisnak** nevezünk, ha a szomszédai egy klikket alkotnak.

Tétel. Minden merev körű gráf vagy teljes, vagy létezik benne kettő, nem szomszédos szimpliciális pont.

Bizonyítás: Csúcsszámra végzett indukcióval bizonyítjuk az állítást. Egy, illetve 2 csúcs esetén az állítás triviális.

Tegyük fel, hogy állítás teljesül minden n-nél kevesebb esetre és legyen G egy n darab csúccsal rendelkező merev körű gráf. Vegyünk egy S minimális vágást G gráfon. Erről már beláttuk, hogy S elemei klikket alkotnak, illeve az így keletkező G_1 és G_2 komponensek is merev körű gráfok lesznek. Továbbá teljesül az is, hogy $S \cup V(G_1)$ csúcsok által indukált részgráf is merev körű, hiszen szintén két merev körű gráfból keletkezett azonos méretű klikkek azonosításaként. Ugyanez teljesül $S \cup V(G_2)$ csúcsok által indukált részgráfra is. Ekkor ezen két gráf az indukciós feltevés miatt vagy teljes, vagy rendelkezik kettő, nem szomszédos szimpliciális csúccsal. Ha teljes, úgy minden csúcsa szimpliciális, beleértve a nem S klikkbeli csúcsokat is. Ha nem teljes, akkor legfeljebb az egyik szimpliciális csúcs lehet S klikkbeli, ellenkező esetben ugyanis szomszédosak lennének. A két gráfban levő nem klikkbeli, szimpliciális pontok jó szimpliciális pontok lesznek az eredeti G gráfban is, mivel S vágás volt.

■

Vegyük észre, hogy amennyiben egy merev körű gráfból eltávolítunk egy szimpliciális csúcsot, úgy a gráf továbbra is merev körű marad, hiszen nem keletkezhet új kör benne. Ez adja az alapötletet egy adott G gráf merev körű tulajdonságát eldöntő, egyszerű algoritmushoz:

1. `Keressünk kettő szimpliciális pontot` G `gráfban.`

2. Amennyiben nincs, úgy a gráf nem merev körű.

3. Amennyiben van, úgy töröljük a gráfból a második szimpliciális pontot.

4. Ismételjük fenti 3 lépést, amíg 1 csúcs nem marad, vagy amíg ki nem derül, hogy a gráf nem merev körű.

3.2. Kapcsolat az intervallumgráfokkal

A merev körű gráfok tulajdonságai teljesülnek az intervallumgráfokra is, ugyanis:

Tétel. Minden intervallumgráf merev körű.

Bizonyítás: Semelyik, legalább 4 hosszú körnek nem létezik intervallum reprezentációja, ugyanis tegyük fel, hogy (a_i, b_i) $i = 1, 2, \ldots, n$ egy adott C_n kör intervallum reprezentációja. Ekkor az alábbi egyenletrendszer érvényesül:

- $b_{i-2} < a_i < b_{i-1} < b_i \quad i = 3, 4, \ldots, n$
- $b_{n-1} < a_1 < b_n < b_1$
- $b_n < a_2 < b_1 < b_2$

De ennek az egyenletrendszernek nincs megoldása. Ezért feszített részgráfként ilyen köröket nem is tartalmazhat intervallumgráf.

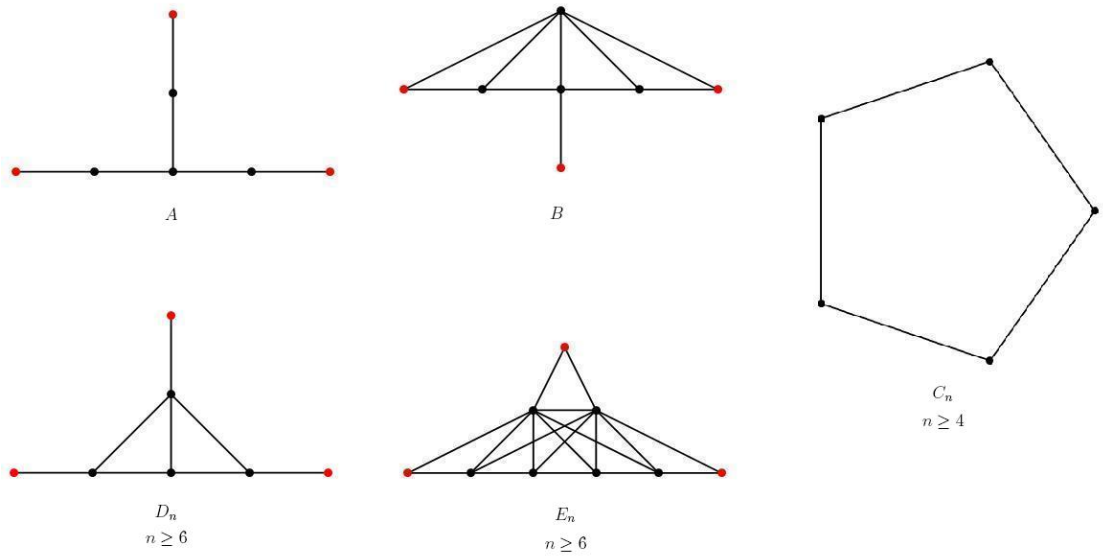
■

A merev körű gráfok azonosítása könnyen megoldható az előző fejezet végén vázolt algoritmussal, mi a helyzet azonban az intervallumgráfokkal?

Az alábbi két tétel Lekkerkerkertől és Bolandtól származik, melyek szükséges és elégséges feltételt adnak az intervallumgráfok azonosítására:

Tétel. Egy G gráf intervallumgráf akkor és csak akkor, ha G nem tartalmaz indukált részgráfként legalább 4 hosszú kört és ha v_1, v_2, v_3 három különböző, nem szomszédos csúcsa G gráfnak, akkor legalább az egyik v_i csúcs szomszédja, vagy eleme az összes, másik két csúcs közt menő útnak. Egy gráfban az ilyen ponthármast **asteroidal triplenek** nevezzük.

Tétel. Egy G gráf intervallumgráf akkor és csak akkor, ha nem tartalmazza feszített részgráfként az alábbi gráfok egyikét sem:



Az ábrán n a csúcsok számát jelöli. Vegyük észre, hogy a fent felsorolt esetek közül, a legalább 4 hosszú kör nem tartalmazása indukált részgráfként pontosan annak felel meg, hogy az intervallumgráf merev körü is. A fennmaradó 4 eset pedig pontosan azoknak a gráfoknak felel meg, melyekben van asteroidal triple.

4. Összegzés

A szakdolgozatban a $PrExt$ algoritmusokkal foglalkoztam. Ismertettem az intervallumgráfok és merev körü gráfok egy néhány tulajdonságát, közöttük azt is, hogy intervallumgráfok esetén az $1-PrExt$ megoldható polinomiális időben, míg az ennél nagyobb esetekre már NP-teljes feladattal van dolgunk. Írtam `Mathematica` -ban egy programot, mely könnyen használható, adott intervallumokra és előszínezett csúcsokra megkonstruálja az előszínezés kiterjesztését, illetve ha ez nem lehetséges, úgy ismerteti ennek okát. Az algoritmust persze lehet továbbfejleszteni. Ismert, hogy az $1 - PrExt$ merev körü gráfok esetében is megoldható polinomiális időben. Számos gráfosztályon azonban még nem ismeretes az eredmény.

5. Függelék

5.1. $1 - PrExt$ algoritmus forráskódja

(*1.Programrész*)

```
<< Combinatorica`
<< GraphUtilities`
SzínSzám = 10;
RandomCsúcsSzám = 17;
Intmax = 100;
Intervallumok = {};

Qeredetiintervallumok = True;
Qrendezetettintervallumok = True;
Qszínszám = True;
Qelőszínezettcsúcsok = True;
QgráfG = True;
Qlegnagyobbklikk = True;
QgráfD = True;
Qmaxfolyamérték = True;
Qmaxfolyam = True;
QgráfSzínezettGgraf = True;
QgráfSzínezettGintervallum = True;
```

(*//*)

(*2.Programrész*)

```
If[Length[Intervallumok] == 0,
Intervallumok = Table[{RandomInteger[Intmax], RandomInteger[Intmax]},
```

```
{i, 1, RandomCsúcsSzám}]];

(*//////////////////////////////////////////////////////*)

(*3.Programrész*)


For[i = 1, i < Length[Intervallumok] + 1, i++,
If[Intervallumok[[i, 2]] < Intervallumok[[i, 1]],
copy = Intervallumok[[i, 2]];
Intervallumok[[i, 2]] = Intervallumok[[i, 1]];
Intervallumok[[i, 1]] = copy;
];
];

Intervallumok = Sort[Intervallumok, #1[[2]] < #2[[2]]&];

RendezettIntervallumok = Table[{0, 0}, {i, 1, Length[Intervallumok]}];
For[i = Length[Intervallumok], i > 0, i- -,
counter = 0;
For[j = 1, j < i, j++,
If[Intervallumok[[i, 1]]>=Intervallumok[[j, 2]], counter++];
];
RendezettIntervallumok[[i, 2]] = i;
RendezettIntervallumok[[i, 1]] = counter;
];


(*//////////////////////////////////////////////////////*)

(*4.Programrész*)
```

```
ElőszínezettCsúcsok = {};
If[Length[ElőszínezettCsúcsok] == 0,
FelhasználtSzínek = RandomInteger[SzínSzám];
RandomTáblázat = Table[i, {i, 1, Length[Intervallumok]}];
ElőszínezettCsúcsok = Table[0, {i, 1, FelhasználtSzínek}];
i = 0;
While[i < FelhasználtSzínek,
i++;
x = (RandomInteger[Length[RandomTáblázat] − 1] + 1);
ElőszínezettCsúcsok[[i]] = RandomTáblázat[[x]];
RandomTáblázat = Drop[RandomTáblázat, {x}];
];
];


(*////////////////////////////////////////////////////////*)

(*5.Programrész*)


SzomszédosságiMátrix = Table[0, {i, 1, Length[Intervallumok]},
{j, 1, Length[Intervallumok]}];
For[i = 1, i < Length[Intervallumok] + 1, i++,
SzomszédosságiMátrix[[i, i]] = 0;
For[j = i + 1, j < Length[Intervallumok] + 1, j++,
If[RendezettIntervallumok[[i, 2]] > RendezettIntervallumok[[j, 1]],
SzomszédosságiMátrix[[i, j]] = 1; SzomszédosságiMátrix[[j, i]] = 1,
SzomszédosságiMátrix[[i, j]] = 0; SzomszédosságiMátrix[[j, i]] = 0];
];
];


(*////////////////////////////////////////////////////////*)
```

```
(*6.Programrész*)


Klikk = MaximumClique[FromAdjacencyMatrix[SzomszédosságiMátrix]];
If[Length[Klikk] > SzínSzám, Túlnagyklikk = True, Túlnagyklikk = False];


(*////////////////////////////////////////////////////////*)

(*7.Programrész*)


If[Túlnagyklikk == False,


i = 0;
EredetiÉlek = RendezettIntervallumok;
While[i < Length[ElőszínezettCsúcsok],
i++;
For[j = 1, j < Length[EredetiÉlek] + 1, j++,
If[EredetiÉlek[[j, 2]] == ElőszínezettCsúcsok[[i]], EredetiÉlek = Delete[EredetiÉlek, j]]
];
];

KisegítőÉlek = Table[{i, i + 1}, {i, 0, Length[Intervallumok]}];

MódosítottÉlek = Table[{0, 0}, {i, 1, Length[ElőszínezettCsúcsok]}];
i = 0;
While[i < Length[ElőszínezettCsúcsok],
i++;
MódosítottÉlek[[i]] = {RendezettIntervallumok[[ElőszínezettCsúcsok[[i]], 1]],
Length[Intervallumok] + 1};
```

```
];

For[i = Length[MódosítottÉlek], i > 1, i- -,
For[j = 1, j < i, j++,
If[MódosítottÉlek[[j, 1]] > MódosítottÉlek[[j + 1, 1]],
copy3 = MódosítottÉlek[[j, 1]];
MódosítottÉlek[[j, 1]] = MódosítottÉlek[[j + 1, 1]];
MódosítottÉlek[[j + 1, 1]] = copy3
];
];
];


(*////////////////////////////////////////////////////*)

(*8.Programrész*)


capacitymatrix = Table[Table[0, {i, 1, Length[Intervallumok] + 2}],
{i, 1, Length[Intervallumok] + 2}];

For[i = 1, i < Length[EredetiÉlek] + 1, i++,
capacitymatrix[[EredetiÉlek[[i, 1]] + 1, EredetiÉlek[[i, 2]] + 1]] =
capacitymatrix[[EredetiÉlek[[i, 1]] + 1, EredetiÉlek[[i, 2]] + 1]] + 1;
];

For[i = 1, i < Length[MódosítottÉlek] + 1, i++,
capacitymatrix[[MódosítottÉlek[[i, 1]] + 1, MódosítottÉlek[[i, 2]] + 1]] =
capacitymatrix[[MódosítottÉlek[[i, 1]] + 1, MódosítottÉlek[[i, 2]] + 1]] + 1;
];

For[i = 1, i < Length[Intervallumok] + 1, i++,
counter = 0;
```

```
For[j = i, j < Length[Intervallumok] + 1, j++,
If[RendezettIntervallumok[[j, 1]] < i, counter++]
];
capacitymatrix[[i, i + 1]] = capacitymatrix[[i, i + 1]] + SzínSzám − counter;
];
capacitymatrix[[Length[Intervallumok] + 1, Length[Intervallumok] + 2]] =
capacitymatrix[[Length[Intervallumok] + 1, Length[Intervallumok] + 2]]+
SzínSzám − Length[ElőszínezettCsúcsok];


(*////////////////////////////////////////////////////////*)

(*9.Programrész*)


Élgráf = {Table[i, {i, 0, Length[Intervallumok] + 1}],
Union[EredetiÉlek, KisegítőÉlek, MódosítottÉlek]};
For[i = 1, i < Length[Élgráf[[2]]] + 1, i++,
Élgráf[[2, i, 1]]++;
Élgráf[[2, i, 2]]++;
];

Élkapacitások = Table[0, {i, 1, Length[Élgráf[[2]]]}];
counter = 1;
For[i = 1, i < Length[Intervallumok] + 3, i++,
For[j = i + 1, j < Length[Intervallumok] + 3, j++,
If[j == (i + 1),
Élkapacitások[[counter]] = capacitymatrix[[i, j]];
counter++;
];
If[capacitymatrix[[i, j]] ≠ 0&&j ≠ (i + 1),
Élkapacitások[[counter]] = capacitymatrix[[i, j]];
counter++;
```

```
];
];
];

Capinfo = Table[0, {i, 1, Length[Élkapacitások]}];
For[i = 1, i < Length[Élkapacitások] + 1, i++,
Capinfo[[i]] = {Élkapacitások[[i]]};
];
Élgráf = Append[Élgráf, Capinfo];


(*////////////////////////////////////////////////////*)

(*10.Programrész*)


 Graph = SetEdgeWeights[FromOrderedPairs[Élgráf[[2]]], Élkapacitások];
MaxFolyam = NetworkFlow[MathematicaGraph, 1,
Length[RendezettIntervallumok] + 2];
Folyam = NetworkFlow[MathematicaGraph, 1,
Length[RendezettIntervallumok] + 2, Edge];

ShowEdgesOptions = Table[{}, {i, 1, Length[Edges[MathematicaGraph]]}];
For[i = 1, i < Length[Edges[MathematicaGraph]] + 1, i++,
ShowEdgesOptions[[i]] = Append[ShowEdgesOptions[[i]],
Edges[MathematicaGraph][[i]]];
ShowEdgesOptions[[i]] = Append[ShowEdgesOptions[[i]],
EdgeLabel → Élkapacitások[[i]]];
];
ShowVerticesOptions = Table[{}, {i, 1, Length[VertexList[MathematicaGraph]]}];
For[i = 1, i < Length[VertexList[MathematicaGraph]] + 1, i++,
ShowVerticesOptions[[i]] = Append[ShowVerticesOptions[[i]],
VertexList[MathematicaGraph][[i]]];
```

```
ShowVerticesOptions[[i]] = Append[ShowVerticesOptions[[i]],
VertexLabel → i − 1];
];

If[MaxFolyam < SzínSzám, Folyamtúlkicsi = True, Folyamtúlkicsi = False];


];


(*////////////////////////////////////////////////////////*)

(*11.Programrész*)


If[Folyamtúlkicsi == False&&Túlnagyklikk == False,



P = Table[{0}, {i, 1, SzínSzám}];
For[i = 1, i < Length[MódosítottÉlek] + 1, i++,
P[[i]] = {MódosítottÉlek[[i]] + 1};
];
For[i = Length[MódosítottÉlek] + 1, i < SzínSzám + 1, i++,
P[[i]] = {{Length[Intervallumok] + 1, Length[Intervallumok] + 2}};
];

Folyamcopy = Folyam;
For[i = 1, i < SzínSzám + 1, i++,
For[j = 1, j < Length[Folyamcopy], j++
If[Folyamcopy[[j, 1]] == P[[i]],
Folyamcopy[[j, 2]]- -;
];
```

```
];
];

k = Length[Folyamcopy];
For[j = 1, j < k + 1, j++,
If[Folyamcopy[[j, 1, 2]] == Length[Intervallumok] + 2,
Folyamcopy = Drop[Folyamcopy, {j}];
j- -;
k- -;
];
];

For[i = 1, i < SzínSzám + 1, i++,
k = P[[i, 1]];
counter = 1;
While[k[[1]] ≠ 1,
If[Folyamcopy[[counter, 1, 2]] == k[[1]]&&Folyamcopy[[counter, 2]] ≠ 0,
Folyamcopy[[counter, 2]]- -;
P[[i]] = Prepend[P[[i]], Folyamcopy[[counter, 1]]];
k = Folyamcopy[[counter, 1]];
counter = 0;
];
counter++;
];
];


(*////////////////////////////////////////////////////////*)

(*12.Programrész*)


MódosítatlanÉlek = Table[0, {i, 1, Length[MódosítottÉlek]}];
```

```
Length[MódosítottÉlek];
For[i = 1, i < Length[MódosítottÉlek] + 1, i++,
MódosítatlanÉlek[[i]] = RendezettIntervallumok[[ElőszínezettCsúcsok[[i]]]] + 1;
];

Pvessző = P;
Módosítatlanélekcopy = MódosítatlanÉlek;
For[i = 1, i < Length[P] + 1, i++,
If[P[[i, Length[P[[i]]], 1]] ≠ Length[Intervallumok] + 1,
j = 1;
While[Pvessző[[i, Length[Pvessző[[i]]], 1]] ≠ Módosítatlanélekcopy[[j, 1]], j++; ];
Pvessző[[i]] = Drop[Pvessző[[i]], {Length[Pvessző[[i]]]}];
Pvessző[[i]] = Append[Pvessző[[i]], Módosítatlanélekcopy[[j]]];
Módosítatlanélekcopy = Drop[Módosítatlanélekcopy, {j}];
];
];


(*//////////////////////////////////////////////////////*)

(*13.Programrész*)


H = Table[{0}, {i, 1, SzínSzám}];
For[i = 1, i < SzínSzám + 1, i++,
H[[i]] = {1, Pvessző[[i, −1, 2]]};
];
H;


(*//////////////////////////////////////////////////////*)

(*14.Programrész*)
```

```
Gvesszőújélek = {};
Gvessző = {};
For[i = 1, i < Length[Intervallumok] + 2, i++,
j = 0;
While[capacitymatrix[[i, i + 1]] ≠ j,
Gvesszőújélek = Append[Gvesszőújélek, {i, i + 1}];
j++;
];
];

Javítóélek = {};
For[i = 1, i < Length[RendezettIntervallumok] + 1, i++,
If[RendezettIntervallumok[[i]] + 1 ≠ {i, i + 1},
Javítóélek = Append[Javítóélek, RendezettIntervallumok[[i]]]];
For[j = 1, j < Length[MódosítatlanÉlek] + 1, j++,
If[(MódosítatlanÉlek[[j]] − 1)=={i − 1, i},
Javítóélek = Prepend[Javítóélek, MódosítatlanÉlek[[j]] − 1]];
];
];

Gvessző = Sort[Join[Gvesszőújélek, Javítóélek + 1]];

For[i = 1, i < Length[Pvessző] + 1, i++,
For[j = 1, j < Length[Pvessző[[i]]] + 1, j++,
counter = 1;
While[Pvessző[[i, j]] ≠ Gvessző[[counter]],
counter++;
];
Gvessző = Drop[Gvessző, {counter}];
];
];
```

```
Gvessző = Join[H, Gvessző];


(*////////////////////////////////////////////////////////*)

(*15.Programrész*)


SzomszédosságiMátrixGvessző = Table[Table[0, {i, 1, Length[Gvessző]}],
{i, 1, Length[Gvessző]}];
For[i = 1, i < Length[Gvessző] + 1, i++,
SzomszédosságiMátrixGvessző[[i, i]] = 0;
For[j = i + 1, j < Length[Gvessző] + 1, j++,
If[Gvessző[[i, 2]] > Gvessző[[j, 1]],
SzomszédosságiMátrixGvessző[[i, j]] = 1; SzomszédosságiMátrixGvessző[[j, i]] = 0,
SzomszédosságiMátrixGvessző[[i, j]] = 0; SzomszédosságiMátrixGvessző[[j, i]] = 0]
];
];


(*////////////////////////////////////////////////////////*)

(*16.Programrész*)


Gvesszőszíneklehetőség = Table[j, {i, 1, Length[Gvessző]}, {j, 1, SzínSzám}];
Gvesszőszínek = Table[0, {i, 1, Length[Gvessző]}];
For[i = 1, i < Length[Gvessző] + 1, i++,
For[j = 1, j < i, j++,
If[SzomszédosságiMátrixGvessző[[j, i]] == 1,
Gvesszőszíneklehetőség[[i]] = Select[Gvesszőszíneklehetőség[[i]], # ≠ Gvesszőszínek[[j]]&];
];
];
```

```
Gvesszőszínek[[i]] = Min[Gvesszőszíneklehetőség[[i]]]
];


(*//////////////////////////////////////////////////////*)

(*17.Programrész*)


Színezés = Table[{RendezettIntervallumok[[i]] + 1, 0},
{i, 1, Length[RendezettIntervallumok]}];

Pm = {};
For[i = 1, i < Length[Pvessző] + 1, i++,
For[j = 1, j < Length[Pvessző[[i]]] + 1, j++,
Pm = Join[Pm, {{Pvessző[[i, j]], Gvesszőszínek[[i]]}}];
];
];


Gm = Table[{Gvessző[[i]], Gvesszőszínek[[i]]}, {i, 1, Length[Gvessző]}];

For[i = 1, i < Length[RendezettIntervallumok] + 1, i++,
j = 1;
While[Színezés[[i, 2]]==0&&j < Length[Pm] + 1,
If[Színezés[[i, 1]] == Pm[[j, 1]], Színezés[[i, 2]] = Pm[[j, 2]]; Pm = Drop[Pm, {j}]; j = 0; ];
j++;
];
j = 1;
While[Színezés[[i, 2]]==0&&j < Length[Gvessző] + 1,
If[Színezés[[i, 1]] == Gm[[j, 1]], Színezés[[i, 2]] = Gm[[j, 2]]; Gm = Drop[Gm, {j}]; j = 0; ];
j++;
];
```

```
];

(*////////////////////////////////////////////////////////*)

(*18.Programrész*)


Színosztályok = Table[{}, {i, 1, SzínSzám}];
For[i = 1, i < Length[Színezés] + 1, i++,
For[j = 1, j < SzínSzám + 1, j++,
If[Színezés[[i, 2]] == j, Színosztályok[[j]] = Append[Színosztályok[[j]], i]; ];
];
];


Színhalmaz = {Black, Green, Red, Blue, Orange, Magenta, Pink, Yellow, Brown, Cyan};
For[j = 1, j < SzínSzám + 1, j++,
Színosztályok[[j]] = Append[Színosztályok[[j]], VertexColor->Színhalmaz[[j]]];
];
Színosztályok;


];


(*////////////////////////////////////////////////////////*)

(*19.Programrész*)


If[Qeredetiintervallumok==True,
Print["Az eredeti intervallumok: "]; Print[Intervallumok]; Print[];
```

```
If[Qrendezetettintervallumok == True,
Print["A szép alakra rendezett intervallumok: "];
Print[RendezettIntervallumok + 1]; Print[]];
If[Qszínszám == True,
Print["A felhasználható színek száma: "]; Print[SzínSzám]; Print[]];
If[Qelőszínezettcsúcsok == True,
Print["A gráf előszínezett csúcsai: "]; Print[Sort[ElőszínezettCsúcsok]]; Print[]];
If[QgráfG==True,
Print["Az intervallumgráf: "]; Print[ShowLabeledGraph[
FromAdjacencyMatrix[SzomszédosságiMátrix]]];
Print[]];
If[Qlegnagyobbklikk == True,
Print["A legnagyobb klikket generáló csúcsok "]; Print[Klikk]; Print[]];
If[QgráfD==True&&Túlnagyklikk == False,
Print["Az G gráfból generált D hálózat: "]; Print[ShowGraph[MathematicaGraph,
Union[ShowVerticesOptions, ShowEdgesOptions], EdgeLabelColor → Red]]; Print[]];
If[Qmaxfolyamérték == True&&Túlnagyklikk == False,
Print["A legnagyobb D hálózatbeli folyam értéke: "]; Print[MaxFolyam]; Print[]];
If[Qmaxfolyam == True&&Túlnagyklikk == False,
Print["A legnagyobb D hálózatbeli folyam egy megvalósítása: "]; Print[P]; Print[]];
If[QgráfSzínezettGgraf==True&&Túlnagyklikk == False&&Folyamtúlkicsi==False,
Print["Az G gráf 1-PrExt színezése gráffal: "];
Print[ShowLabeledGraph[SetGraphOptions[FromAdjacencyMatrix[
SzomszédosságiMátrix], Színosztályok]]]];
If[QgráfSzínezettGintervallum==True&&
Túlnagyklikk == False&&Folyamtúlkicsi==False,
Print["Az G gráf 1-PrExt színezése intervallumokkal: "];
Print[Színezés]];
If[Túlnagyklikk == True, Print["Találtam egy túl nagy klikket:"];
Print[Klikk];
Print["A színezés nem terjeszthető ki!"]];
If[Folyamtúlkicsi == True,
Print["Nincs elég nagy folyam, a színezés nem terjeszthető ki!"]];
```

5.2. Egy konkrét példa bemutatása

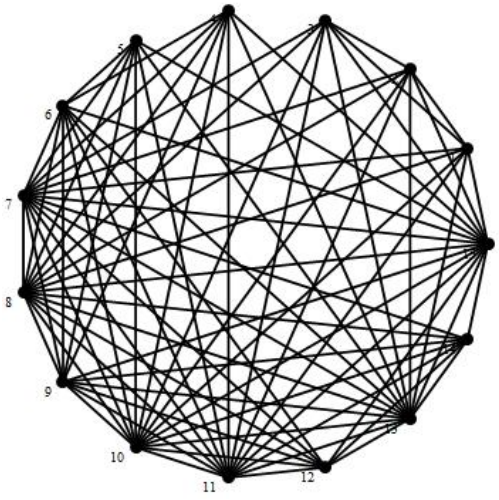
Az eredeti intervallumok:
$\{\{27,39\},\{1,46\},\{15,47\},\{55,56\},\{57,57\},\{47,69\},\{6,70\},\{38,76\},$
{50,89},{20,89},{20,90},{57,91},{11,96},{69,99},{32,100}}

A szép alakra rendezett intervallumok:
$\{\{1,2\},\{1,3\},\{1,4\},\{4,5\},\{5,6\},\{4,7\},\{1,8\},\{1,9\},\{4,10\},$
{1,11},{1,12},{6,13},{1,14},{7,15},{1,16}}

A felhasználható színek száma:
10

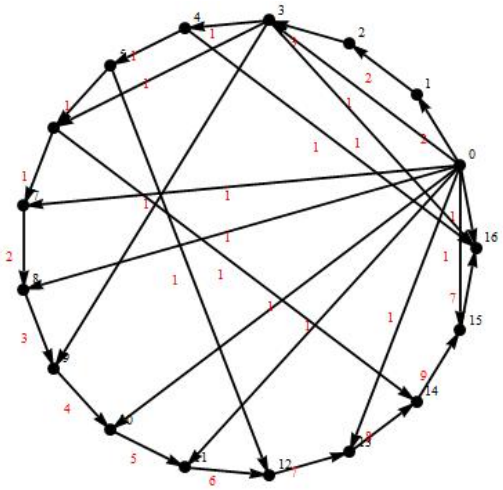
A gráf előszínezett csúcsai:
$\{2,4,5\}$

Az intervallumgráf:



A legnagyobb klikket generáló csúcsok
$\{1,2,3,7,8,10,11,13,15\}$

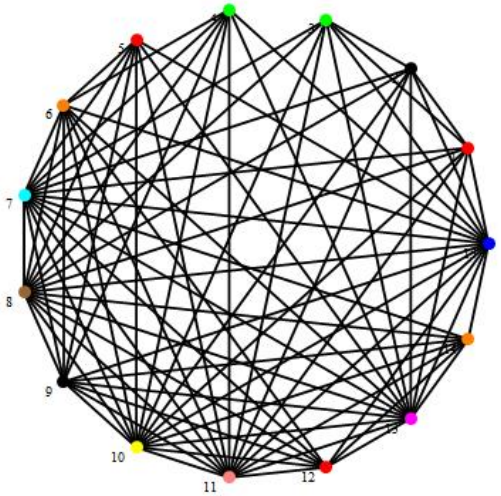
Az G gráfból generált D hálózat:



A legnagyobb D hálózatbeli folyam értéke:
10

A legnagyobb D hálózatbeli folyam egy megvalósítása:
{{{1, 17}}
{{1,4},{4,17}}
{{1,2},{2,3},{3,4},{4,5},{5,17}}
{{1,16},{16,17}}
{{1,2},{2,3},{3,4},{4,7},{7,15},{15,16},{16,17}}
{{1,14},{14,15},{15,16},{16,17}}
{{1,12},{12,13},{13,14},{14,15},{15,16},{16,17}}
{{1,11},{11,12},{12,13},{13,14},{14,15},{15,16},{16,17}}
{{1,9},{9,10},{10,11},{11,12},{12,13},{13,14},{14,15},{15,16},{16,17}},
{{1,8},{8,9},{9,10},{10,11},{11,12},{12,13},{13,14},{14,15},{15,16},{16,17}}}

Az G gráf 1-PrExt színezése gráffal:



Az G gráf 1-PrExt színezése intervallumokkal:
$\{\{\{1,2\},3\},\{\{1,3\},1\},\{\{1,4\},2\},\{\{4,5\},2\},\{\{5,6\},3\},\{\{4,7\},5\},$
{{1,8},10},{{1,9},9},{{4,10},1},{{1,11},8},{{1,12},7},{{6,13},3},
{{1,14},6},{{7,15},5},{{1,16},4}}

5.3. Felhasznált irodalom

- Katona Gy. Y., Recski A., Szabó Cs.: A számítástudomány alapjai (Typotex kiadó, Budapest, 2002)
- M. Biró, M. Hujter, Zs. Tuza: Precoloring extension. I. Interval graphs (Discrete Mathematics 100 (1992) 267 – 279.)
- G. Chartrand, P. Zhang: Chromatic graph theory (CRC Press, 2009)
- D. Marx: Graph Coloring with Local and Global Constraints (Budapest, 2004)
- C.G. Lekkerkerker and J.C. Boland: Representation of a finite graph by a set of intervals on the real line (Fundamenta Math. 51 (1962) 45 – 64.)
- K. Cameron, C. T. Hoàng, B. Lévêque: Asteroids in rooted and directed path graphs (Discrete Mathematics 32 (2009))