Konvex testek topológiai ősének meghatározása

Szekeres Zoltán

Konzulens: Kápolnai Richárd, Irányítástechnika és Informatika Tanszék

1. Bevezetés

A dolgozatomban egy elméleti eredményt ültettem át a gyakorlatba, azáltal, hogy implementáltam egy algoritmust, amely konvex testek topológiai ősét határozza meg. Az ős meghatározáshoz felhasználom, hogy egy konvex testet elsődleges egyensúlyi osztályba sorolhatunk úgy, hogy egy (S, U) számpárt rendelünk hozzá, melynek első tagja a test stabil, a második a test instabil egyensúlyi pontjainak száma [1]. Például egy kockának hat stabil és nyolc instabil pontja van, ezért a kocka elsődleges egyensúlyi osztálya (6, 8). Ezen belül figyelembe vesszük az egyensúlyi pontok topológiáját is, amit a felület Morse–Smale-gráfja határoz meg. Ily módon a test másodlagos egyensúlyi osztályát egy gráf határozza meg [2]. A gráf csúcshalmazát a test egyensúlyi pontjai (stabil, instabil és nyeregpontok) alkotják. Az éleket az egyensúlyi pontok közötti bizonyos pályák alapján húzzuk be.

Az algoritmust Kolumbusz Kristóf tiszteletére nevezték el. Aki a legenda szerint egy vacsorán, amit a tiszteletére rendeztek, megkérdezte a résztvevőktől, hogy képesek-e egy tojást a csúcsán megállítani. Miután többen is próbálkoztak, sikertelenül, arra a következtetésre jutottak, hogy ez nem lehetséges. Kolumbusz viszont elhelyezte a tojást az asztalon, oly módon, hogy a csúcsánál megtörte, deformálta kicsit a tojás héját, úgy, hogy a kialakult síkon már megállt a tojás. A bemutatót követően Kolumbusz kijelentette, hogy ez a világ legegyszerűbb dolga, bárki képes utána csinálni, miután megmutatta, hogyan kell.

Kolumbusz-algoritmus lépései ismert geometriai transzformációk, amelyek úgy módosítanak egy testet, hogy a konvex test stabil egyensúlyi pontjainak vagy instabil egyensúlyi pontjainak számát növelik eggyel, egy egyensúlyi pont lokális környezetének módosításával [1]. E transzformációk elméleti jelentőséggel bírnak, segítségükkel bizonyítható, bármilyen (S, U) elsődleges osztálybeli konvex test geometriája létrehozható ilyen lépések sorozatával a Gömböcből kiindulva, amely az (1, 1) elsődleges osztályba tartozik. Ha a Gömböcből előállítható minden elsődleges osztály, vajon minden lehetséges másodlagos osztály is? Nem, vannak olyan másodlagos osztályok, amelyek, sem a Gömböcből, sem más testből nem állíthatók elő. Ezek a topológiai ősök, röviden ősök. A Gömböcön kívül vannak más kitüntetett ősök is. Ilyen ős pl. a tetraéder.

Ebben a dolgozatban a Kolumbusz-algoritmussal elért elméleti eredményeket ültettem át a gyakorlatba és vizsgáltam meg erről az oldalról, mint az elméleti eredmény kísérleti alkalmazása.

Az algoritmus megvalósított implementációja komplexitását tekintve a lépésszám a bemenet négyzetével arányos, viszont a feladathoz megfelelőbben választott adatszerkezet felhasználásával sikerült belátnom, hogy a feladat implementálása megoldható oly módon, hogy a lépésszám lineáris legyen a bemenet függvényében.

Az algoritmust lefuttattam a Szilárdságtani és Tartószerkezeti Tanszék munkatársai által 3D scannerrel beolvasott 98 természetes kavics gráfjain [3], valamint rendelkezésre állt az összes lehetséges gráf S+U<=10 méretkorlátig, amelyekre szintén lefuttatva az algoritmust, az

eredmény az volt, hogy a gráfok több, mint 99,9 % -a egy topológiai őstől származtatható, a Gömböctől. Egy másik ős, amely többször előfordult az a már említett egyik kitüntetett ős, a tetraéder. Ez a második legkisebb csúcsszámú lehetséges ős. A futtatás során összesen több mint 80 ezer gráfra kellett meghatároznom az őst.

A dolgozatom 2. és 3. fejezetében, az algoritmus megértéséhez szükséges fogalmakat és háttér információkat ismertetem. A 4. fejezetben bemutatom a megvalósított implementáció működését, meghatározom a komplexitását. Az 5. fejezetben bemutatom, hogyan lehet az algoritmus lépésszámán csökkenteni, majd az utolsó fejezetben a tesztelés eredményeit írom le.

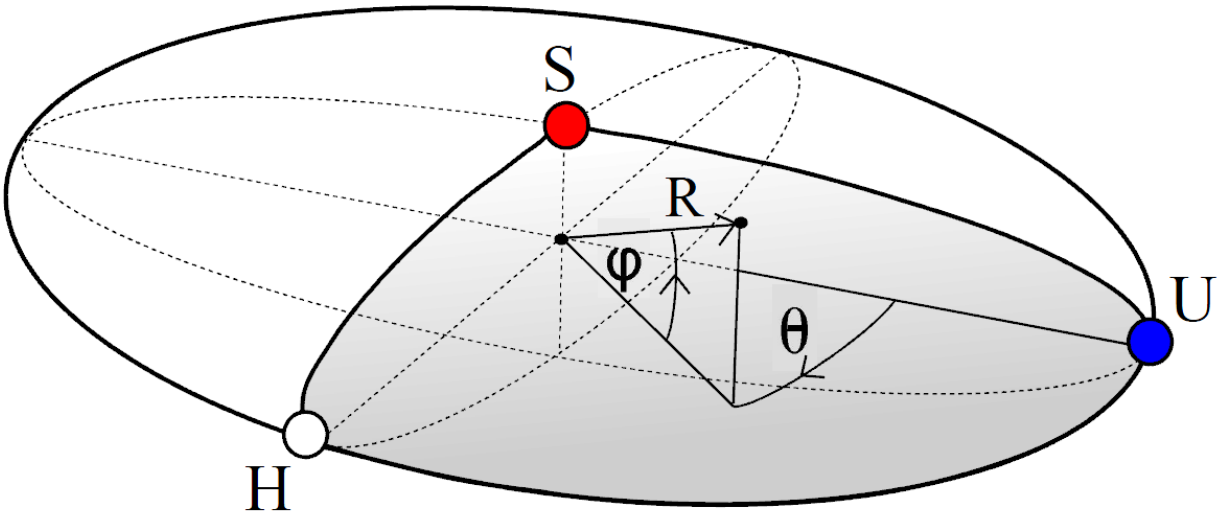
2. Konvex test egyensúlyi osztályozása

Ebben a fejezetben konvex testek osztályozásához szükséges fogalmakat és háttér információkat írom le. Az osztályozás ismertetését először az elsődleges, majd a másodlagos osztály ismertetésével

2.1.Konvex test egyensúlyi pontjainak meghatározása

A háromdimenziós, homogén, konvex testek egyensúlyi pontjaik segítségével egy természetes osztályozási rendszert vezethetünk be. Az osztályozást tovább bővíthetjük, azáltal, hogy ezen, elsődleges osztályozáson belül, másodlagos osztályokba soroljuk. Ezeket az osztályokat a test geometriai tulajdonságai határozzák meg. Az elsődleges osztályt a test egyensúlyi (stabil és instabil) pontjaik száma alapján határozhatjuk meg, míg a másodlagos osztályokat a test Morse-Smale-gráfja definiálja [4].

Az elsődleges osztályozás ismertetéshez vegyük, az R(θ, φ) magasságfüggvényt, amely megmondja minden irány esetén, hogy a test felszíne milyen messze van a test tömegközéppontjától. Az elsődleges osztályozáshoz továbbá szükségünk van ezen magasságfüggvény gradiensére, amely megadja a minden irány esetében a magasságfüggvény legnagyobb meredekségének irányát és nagyságát. A gradiens mező zérushelyei határozzák meg az egyensúlyi pontokat. Ennek generikus, konvex esetben három típusa fordulhat elő. A minimumhelyeken lesznek a test stabil egyensúlyi pontjai, a maximumhelyeken a test egyensúlyi instabil pontja, valamint a harmadik előforduló típus a nyeregpont. A nyeregpontok olyan pontok a gradiens mezőn, amelyet egyik irányból közelítve lokális minimumot kapunk, viszont egy arra merőleges irányból közelítve lokális maximumot. Például az 1. ábrán látható az ellipszoid, jelölve rajta az egyik stabil egyensúlyi pontját, az egyik instabil egyensúlyi pontját, illetve egyik nyeregpontját. Az ábrán S-sel van jelölve a stabil egyensúlyi pont, U-val az instabil egyensúlyi pont és H-val a nyeregpont. A stabil egyensúlyi pontok számát jelölje S, az instabilakét U, a nyeregpontokét H. Ekkor a Poincaré-Hopf tétel [1] értelmében, mivel háromdimenziós testekről van szó, $|S| + |U| - |H| = 2$. Így az (S, U) számpárossal egyértelműen azonosíthatjuk az egyensúlyi pontok számát, tehát ez a számpár egy elsődleges egyensúlyi osztályt határoz meg.

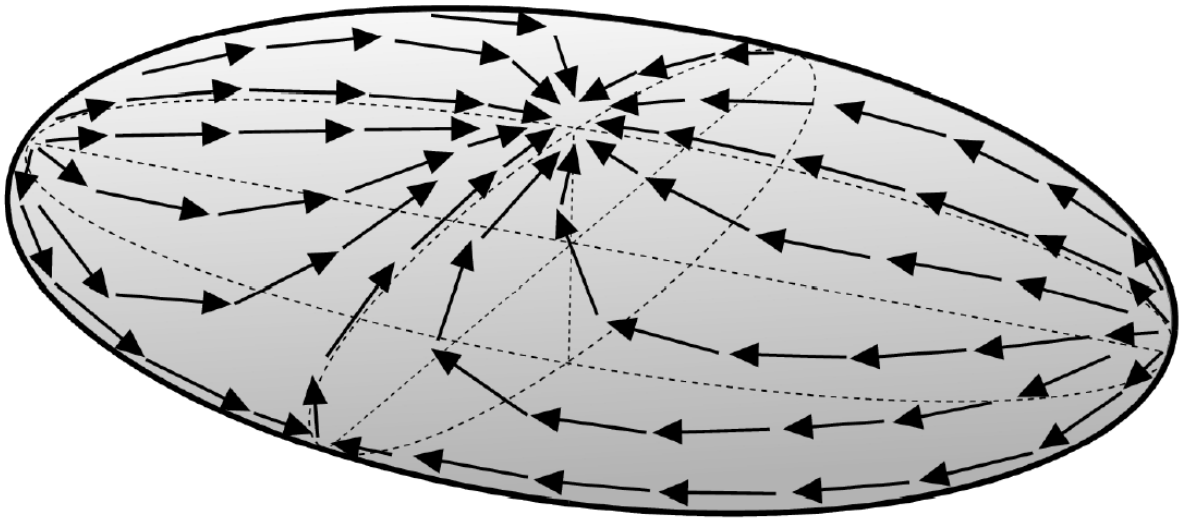


1. ábra Ellipszoid felülete

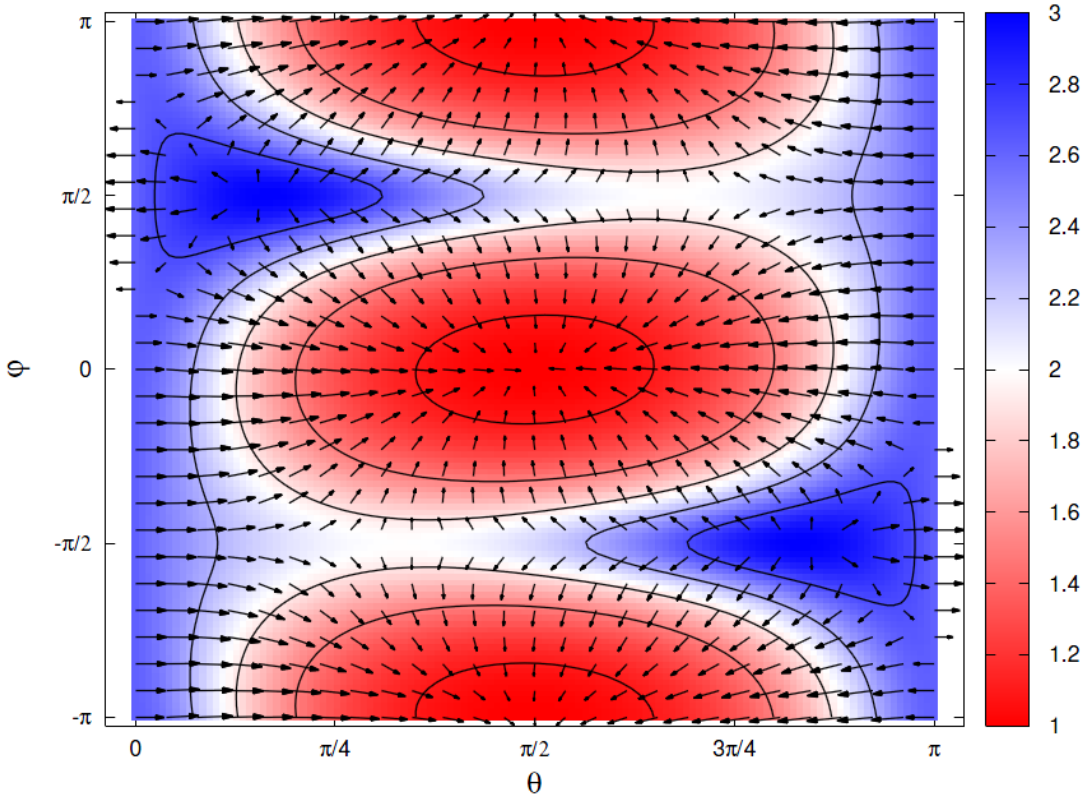
Ismert, hogy minden pozitív egész (S, U) számpárhoz létezik legalább egy olyan test, melynek S stabil és U instabil egyensúlyi pontja van, szóval nincs üres (S, U) osztály [1]. Ennek az állításnak a helyességnek szemléltetése két lépésből áll. Tudjuk, hogy létezik test az (1, 1) osztályban. Ezt bizonyítja a Gömböc létezése, amelynek egyetlen stabil és egy instabil pontja van. Az állítás második részének belátásához szükséges olyan geometriai transzformáció, amely bármely testet lokális környezetében módosítja, így egy (S, U) osztályú testből egy (S+1, U) vagy egy (S, U+1) osztályú testet hoz létre. Ilyen transzformációra példa a Kolumbusz transzformáció.

2.2.Egyensúlyi pontok alapján gráf elkészítése

Az egyensúlyi pontok száma szükséges volt ahhoz, hogy egy konvex testet elsődleges osztályba soroljunk, viszont egy osztályon belül nagyszámú különböző elrendezésű test található. Ezeket a testeket a másodlagos osztályozás segítségével különböztetjük meg egymástól, mégpedig úgy, hogy a testeket gráfokként reprezentáljuk az R(θ, φ) magasságfüggvény által definiált Morse-Smale-komplex alapján [4]. A gráf csúcsait a már definiált egyensúlyi pontok adják, míg az élei meghatározásához szükséges a gradiens-mező bizonyos pályáit tekinteni. Egy pályán egy olyan útvonalat értünk, amelynek mindenütt érintője a gradiens. Azok az izolált pályák, amelyek egyik végén nyeregpont található és másik végén egy stabil egyensúlyi pont vagy egy instabil egyensúlyi pont van, azok jól definiáltak. A másodlagos osztályok definiálásához ezeket az izolált pályákat vesszük figyelembe. A 2. ábrán látható a gradiens mező a felületen, illetve a 3. ábrán látható a gradiens mező a síkban.



2. ábra Magasságfüggvény gradiensmezője a felületen (forrás [2])



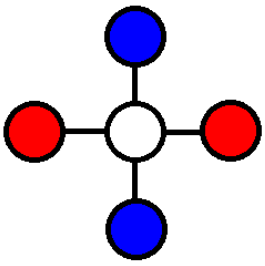
3. ábra Magasságfüggvény gradiendmezője a síkon (forrás [2])

A másodlagos osztályozáskor egy gráfként reprezentáljuk a testet, amely gráfnak az csúcsai, a test egyensúlyi pontjai és élei a bizonyos izolált pályák a gradiens-mezőn. Az így kapott gráf tulajdonságai az alábbiak.

- A gráf összefüggő.
- Csúcsai három színosztályba sorolhatóak, mivel azonos típusú egyensúlyi pontok között nem vezet a gradiens-mezőn pálya, így a kapott gráfon sem vezet azonos típusú

egyensúlyi pontok között él. Rajzban a stabil, instabil és nyeregpontokat rendre piros, kék és fehér színnel jelölöm, és nem lesz két azonos színű pontot összekötő él.

- A nyeregpontok mindig negyedfokúak, azaz a nyeregpontokra illeszkedő élek száma minden esetben négy. A szomszédos csúcsok egyenlően oszlanak el stabil és instabil pontok között, tehát két stabil és két instabil pont alkotja a szomszédjait. Körbejárva a nyeregpont szomszédjait, a stabil egyensúlyi pontok és az instabil egyensúlyi pontok váltakozva fordulnak elő, tehát nem lehet olyan eset, hogy a pl. két stabil pont egymás mellett van. Egy helyes lehetőséget vázol a 4. ábra. Előfordulhat, hogy valamelyik ponthoz kettő él is vezet, azonban a nyeregpont fokszáma nem lehet négytől eltérő, tehát ilyen esetben, csak három ponttal van összekötve.
- A kapott gráf síkgráf. Mivel a gradiens mezőn az izolált pályák nem metszik egymást, így a síkgráf esetében sem fordulhat elő, hogy valamelyik két él metszi egymást.
- A gráf négyszögelt. Egy síkgráf a síkot tartományokra osztja. A négyszögeltség annyit jelent, hogy minden tartományt négy él határol, beleértve a külső, végtelen tartományt is. Továbbá minden tartományra igaz, hogy valamelyik körüljárási irány szerint megyünk sorba a pontok akkor nyeregpont, stabil pont, nyeregpont, instabil pont sorrendet találunk.



4. ábra Stabil és instabil pontok lehetséges elrendezése

Gyakorlati okok miatt, a négyszögelt gráfból néhány új él behúzásával egy háromszögelt gráfot készítünk. Ez a háromszögelt gráf nem tartalmaz több információt, mint a négyszögelt gráf, viszont a topológiai ős keresésekor megkönnyíti a helyzetet. A háromszögeltség fogalma hasonló a négyszögeltséghez, csak a tartományokat négy él helyett, három él határolja. Az új éleket a stabil egyensúlyi és az instabil egyensúlyi pontok közé húzzuk be. Ennek menete a következő. Vesszük a síkgráf összes tartományát, és minden tartomány esetén összekötjük a stabil egyensúlyi pontot az instabil egyensúlyi ponttal, így a négyszög tartományból kaptunk két háromszög tartományt. Az 4. ábra bal oldalán látható az eredeti négyszögelt esetet. A jobb oldalt már a háromszögelt eset látható.

5. ábra A gráf háromszögeltségítése

A fenti tulajdonságokból már automatikusan következik a Poincaré-Hopf tétel által kimondott $|S| + |U| - |H| = 2$ összefüggést. Ezek a tulajdonságok fontos elemei a megvalósításnak.

Ahhoz, hogy az összefüggés helyességét ellenőrízzük szükségünk van az Euler-formulára, amely összefüggő, síkgráfok esetén kimondja, hogy $n - e + t = 2$ összefüggés igaz, ahol n a gráf csúcsainak számát, e a gráf éleinek számát, t pedig a gráf tartományainak számát (beleértve a külső végtelen tartományt is) jelenti [5]. Az ellenőrzést négyszögelt gráfok esetén végzem el. Tudjuk, hogy minden él a gráfban pontosan egy nyeregponthoz kapcsolódik. Mivel minden nyeregpontnak négy a fokszáma és a nyeregpontok száma $|H|$, ezért $e = 4 * |H|$. Továbbá minden tartományt négy él határol és minden él két tartomány határa ezért $4 * t = 2 * e = 2 * 4 * |H| = 8 * |H|$, tehát $t = 2 * |H|$. Most ha felírjuk az Euler formulát, akkor a következő egyenletet kapjuk: $n - 4 * |H| + 2 * |H| = 2$, amit átrendezve $n = 2 * |H| + 2$ –re jutunk. Felhasználva, hogy $|S| + |U| + |H| = n$, kapjuk, hogy $|S| + |U| + |H| = 2 * |H| + 2$, amiből már látszik az összefüggés helyessége.

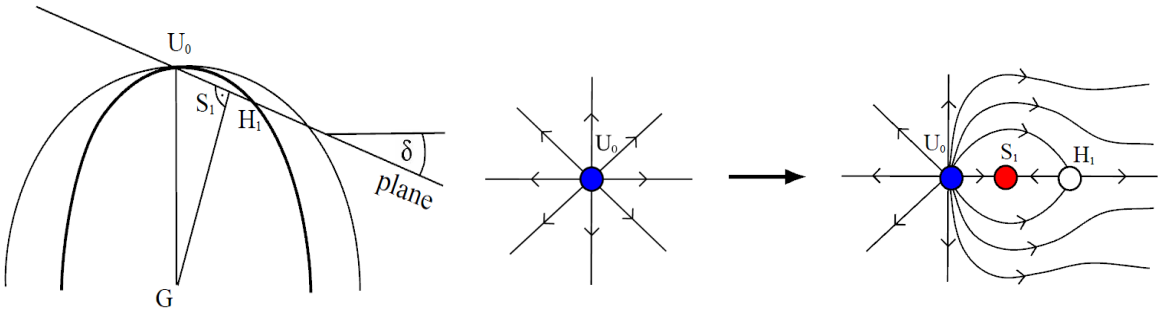
3. A Kolumbusz algoritmus

A Kolumbusz-algoritmus jelentősége, az a fontos elméleti eredmény, hogy Gömböcből kiindulva bármely elsődleges osztályhoz konstruálhatunk geometriát.

3.1. Az algoritmus működése

Az algoritmus úgy működik, hogy a testet egy stabil egyensúlyi pont vagy egy instabil egyensúlyi pont lokális környezetében deformálja, és így egy egyensúlyi pont mellett egy új instabil egyensúlyi pont és nyeregpont vagy egy stabil egyensúlyi pont és egy nyeregpont keletkezik. Belátható hogy ez mindig megvalósítható (bármelyik konvex testhez vehetünk fel ilyen módon újabb pontokat), viszont a lépést visszafelé már nem minden esetben tudjuk elvégezni. Ez kitüntetett testek esetében fordul elő. Ezek a topológia ősök. Ezt a tulajdonságot később felhasználom a megvalósításnál, mivel ilyen esetben kell az algoritmusnak leállnia. Egy ponton biztosan teljesülni fog, mivel a Gömböc osztályánál tovább nem mehetünk visszafelé.

Az alábbi két ábrán látható, hogy a transzformáció, hogy módosítja a felületet.



6.2 ábra Két egyensúlyi pont jelenik meg a felületen (forrás [2])

6.1. ábra Felszín levágása síkkal (forrás [2])

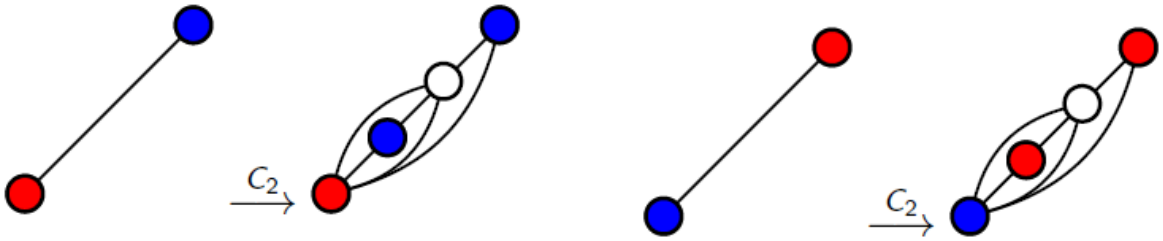
3.2. A redukció jelentése gráfokra nézve

A Kolumbusz-algoritmust azonban nem csak a geometrián, hanem a gráfon (másodlagos egyensúlyi osztályon) is értelmezzük. A topológiai ősök a gráfok esetében azok, amelyek nem állíthatóak elő Kolumbusz-algoritmus segítségével egy másik gráfból. Az eljárás során ilyen topológiai ősöket kerestem, amelyeken a fentebbi lépéseket már nem lehet visszafelé elvégezni. A lépést elvégezve a gráf csúcsainak száma növekszik, ezt nevezzük expanziónak. Az inverzét, amikor csökkentjük a pontok számát pedig redukciónak (ahogy szokás [6]).

A gráfokon elvégezhető lépések a Gömböc esetét kivéve, minden más gráf esetén megegyezik. Mivel az algoritmus csak egy pont lokális környezetében deformálja a testet,

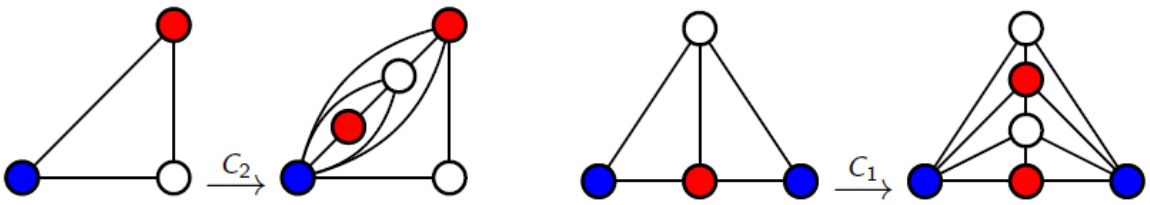
ezért a gráfon is egy lépés vizsgálatához elegendő egy pont és annak lokális környezetének vizsgálata, mert a többi nem változik.

Először ismertetem, hogy a Gömböc esetében, milyen lehetséges expanziók lehetségesek, majd ezt követően a többi gráfra, általánosan. A Gömböc esete azért kivételes, mert a gráfja mindössze kettő csúcsból áll, ebből következőleg nem tartalmaz nyeregpontot, valamint a Gömböc gráfja nem háromszögelt (vagy ha a stabil és instabil pontok közötti éleket nem tekintjük, akkor nem négyszögelt). A 7. ábra a Gömböc elvégezhető kétféle lehetséges expanziót mutatja be.

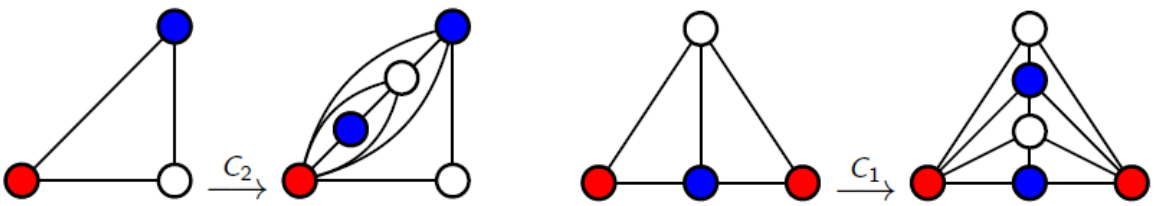


7. ábra A Gömböcön végrehajtható lehetséges expanziók

A többi esetben nem kell a teljes gráfot figyelnünk, elegendő legfeljebb hat érintett csúcsot figyelembe venni. Ezek az alábbi ábra szerint változnak egy expanziók esetén. Az expanziót minden esetben végre lehet hajtani. Kétféle lehetséges expanzió létezik, viszont a stabil és instabil pontok felcserélésével hasonló expanziót kapunk, ezért az ábrán összesen négy esetet vázolok fel. A 8. ábra mutatja az expanziókat, amikor új stabil pont keletkezik és a 9. ábra mutatja azokat, amikor új instabil pont keletkezik. Az ábra csak azokat a pontokat jelzi a gráfon, amelyek szerepet játszanak az expanzióban. Az ábrázolt gráfon belül több él nem futhat, viszont kívülről a pontokhoz kapcsolódhatnak még élek.



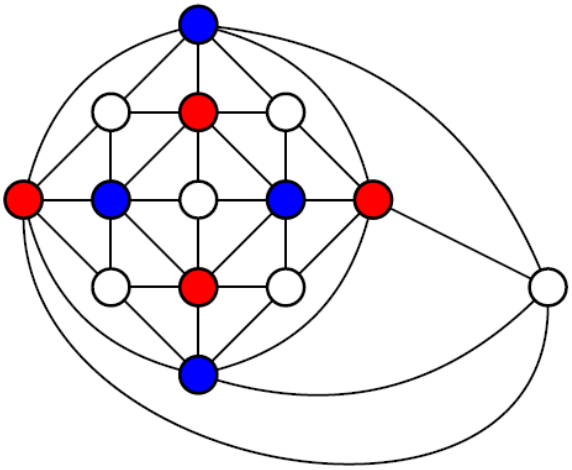
8. ábra Stabil egyensúlyi jelenik meg



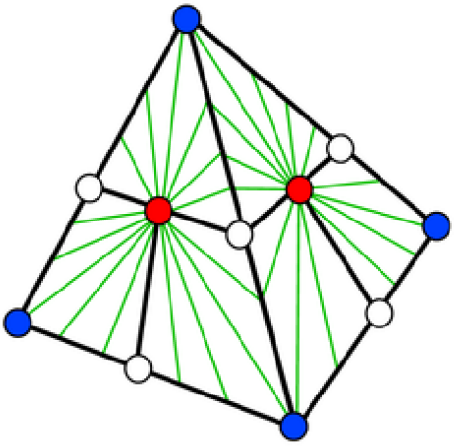
9. ábra Instabil egyensúlyi jelenik meg

Látható, hogy a C_2 expanzió esetén egy másod fokú csúcs keletkezett, a C_1 expanzió esetén pedig egy negyedfokú, tehát a redukció elvégzésének szükséges feltétele, hogy a gráf tartalmazzon legalább egyet a kettő közül. Belátható, hogy ez a feltétel elégséges is [2].

A Gömböc utáni legkisebb fokszámú topológiai ős a tetraéder. Ennek a gráfjának tizennégy csúcsa van és minden stabil és instabil pontjának fokszáma hat, ebből következik, hogy nem hajtható rajta végre Kolumbusz redukció. A tetraéder egy minimálpoliéder, ami egy olyan poliéder, amely minden lapja tartalmaz egy stabil egyensúlyi pontot és minden csúcsa egy instabil egyensúlyi pont. Általánosan minden minimálpoliéderre igaz, hogy nem hajtható rajtuk végre Kolumbusz-redukció, tehát ősök. Ennek a bizonyítása elég egyszerű, csupán azt kell látni, hogy minden lapjára (stabil pont) illeszkedik minimum három-három csúcs és él, valamint minden csúcsra (instabil pont) illeszkedik legalább három-három lap és él, így ennek a testnek a gráfján, minden S és U pontjának fokszáma minimum hat. A 10. ábrán látható a tetraéder gráfja. Látható, hogy minden stabil és instabil pontjának fokszáma hat.



10. ábra A tetraéder gráfja a síkon



11. ábra A tetraéder egyensúlyi gráfja a felületén (forrás [2])

4. Az implementáció működése

A Kolumbusz-algoritmus megadja, hogy a redukció egyes lépési mit foglalnak magukba, viszont azt, hogy az algoritmus implementálása során, hogyan járjak el és milyen adatszerkezeteket használjak, nekem kellett eldönteni. Ebben a fejezetben ismertetem az választott adatszerkezetek és a redukció implementálásnak lépéseit.

4.1. A megvalósítás lépései

A feladatom az volt, hogy adott konvex testek gráfja esetén határozzam meg a topológiai őseiket. Ennek elvégzéséhez adott volt számomra 98 darab a Szilárdságtani és Tartószerkezeti Tanszék munkatársai által 3D szkennerrel beolvasott természetes kavics gráfja. Ezeknek a gráfoknak a szomszédossági mátrixai [5] vannak megadva és a korábban már említett négyszögelt gráfot írtak le, tehát csak a nyeregpontok szomszédjai voltak adottak.

A 11. ábrán egy természetes kavics 3D szkenner beolvasott szomszédossági mátrixa látható. A bemenet tartalmazza a csúcsok térbeli koordinátáit is. A bemenetben a sorok jelentik a nyeregpontokat. Az oszlopok a stabil illetve instabil egyensúlyi pontokat. Egy nyeregpont és egy stabil vagy instabil egyensúlyi pont között akkor fut él, ha a szomszédossági mátrixban, az adott sor és oszlop által meghatározott cellában, zérustól különböző érték van. Ha ez az érték pozitív akkor a nyeregpont egy stabil egyensúlyi ponttal van összekötve, ha negatív akkor egy instabil egyensúlyi ponttal.

*	*	x:	-0.126099	0.129267	-9.66829	9.79159
*	*	y:	0.317409	-0.0741957	0.502603	-1.07662
x	y	z:	2.96359	-2.75857	-0.372219	0.30037
-0.711213	-6.3559	0.00272804	0.47805907	0.5132587	-0.46427493	-0.4867413
0.406614	6.55867	-0.416934	0.50502295	0.54022258	-0.43731105	-0.45977742

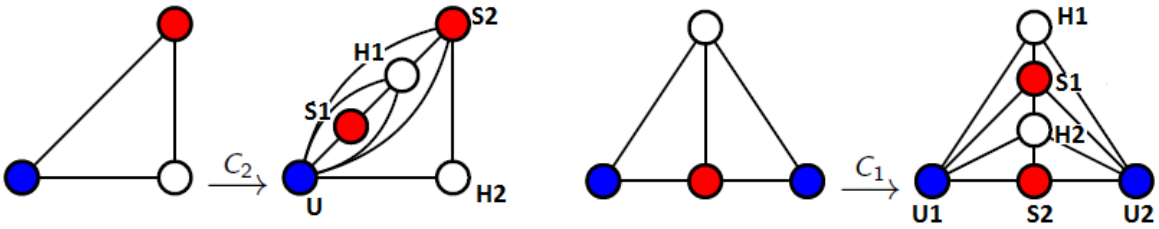
12. ábra Példa bemenet

Első lépésként a beolvasással együtt be kellett húznom az éleket a stabil és instabil pontok közé, mivel ezek nem voltak megadva. Ezekez az éleket külön mátrixban tároltam, így elkülönültek a megadott adatoktól és egyszerűbb volt a kezelésük is. Ehhez szükséges volt megtalálnom minden négyszöget és azokba behúzni a megfelelő éleket, hogy háromszögelt legyen a gráf. Ennek megvalósítása során megkerestem, minden nyeregpont esetében, hogy melyik két stabil és melyik két instabil ponttal van összekötve és ezt követően, minden lehetséges módon élet húztam a megtalált csúcsok közé, arra ügyelve, hogy azonos típusú csúcsokat ne kössek össze. Viszont minden tartományban kettő nyeregpont található, tehát

minden újabb élet pontosan kétszer találtam meg, ezért a függvény végén le kellett osztanom a szomszédossági mátrix összes elemét kettővel. Ezt követően megvolt minden adatom, ahhoz hogy elkezdjem a topológiai ős keresését.

A kereséshez a Kolumbusz-redukciókat használtam, így lépésenként kettővel csökkentettem a csúcsok számát. Tudjuk, hogy a redukció megvalósításának szükséges és elégséges feltétele, hogy a gráfban legyen egy másodfokú pont vagy egy negyedfokú, ami nem nyeregpont. Amíg ez a feltétel fennáll, addig biztos, hogy a gráfon lehet még további redukciót elvégezni, azonban, ha nem találok ilyet, akkor szintén biztos, hogy megtaláltam a keresett topológiai őst.

Az elméletet követve, minden ciklusban kerestem egy ilyen csúcsot. A keresés gyorsítására érdekében a beolvasást követően megszámoltam minden csúcs fokszámát és eltároltam egy tömbben, így a két mátrix helyett, csak egy tömb elemein kell végigmennem egy megfelelő fokszámú csúcs megtalálásához. Majd minden redukciót követően frissítem a tömböt és mivel egy redukció legfeljebb négy nem nyeregpontot érint. Ezért a tömbön is legfeljebb négy elemet kell frissíteni redukciónként. Ha megvan már a redukció kulcspontja (a másodfokú vagy negyedfokú pont, amit kerestem) akkor meg kellett találni, a többi, a redukciót érintő pontot. Itt is élek a feltételezéssel, hogy a megtalált pont stabil, ami magát a redukció lépéseit nem befolyásolja, viszont megkönnyíti a leírását. A csúcsok az alábbi módon lettek elnevezve.



13. ábra A csúcsok elnevezése

Ennek a két esete:

- Az első eset, ha negyedfokú pontot találok. Ebben az esetben összesen hat pontot érint a lépés, amiből öt pont már meg is van ($S1$ és a négy szomszédja). A hatodik pont megtalálásához csupán a szomszédossági mátrixban kell megkeresnem azt a pontot, amelyik illeszkedik az ábrán látható mintára, tehát szomszédos $H2$-vel valamint $U1$-gyel és $U2$-vel. Ezt követően a redukció által meghatározott éleket törölnöm kell a szomszédossági mátrixból, valamint a fokszámokat tároló tömbből és csökkenteni a fokszámokat a meghatározott értékekkel. Az utolsó lépés egy új él felvétele $H1$ és $S2$ között.
- Ha másodfokú pontot találok, akkor a következőképpen járok el. Először megnézem $S1$ szomszédjait, amelyek szintén szükségesek a redukcióhoz. Mivel a redukcióban öt csúcs vesz részt, így még két másikat kell megtalálnom. Azonban $H2$-t valójában nem is kell megtalálnom, mivel $S2$-vel és U-val össze van kötve, ezért a gráf háromszögeltsége miatt léteznie kell egy nyeregpontnak, amelyik össze van kötve $S2$-

vel és U-val. Ráadásul $H2$-nek nem változtatjuk sem a fokszámát, sem pedig a szomszédjait. $S2$ meghatározása sem jelent gondot, hiszen mindössze a $H1$-gyel összekötött másik stabil pontot kell megkeresni a szomszédossági mátrixban.

A keresés főciklusában ez a két eset váltogatja egymást. A leállásnak két feltétele lehetséges. Az első, ha már nincs sem negyedfokú sem másodfokú pont. Ilyenkor egy olyan topológiai őst találtam, amelyik nem a Gömböc. Előfordulhat, hogy találok a még másodfokú csúcsot a gráfban, viszont ekkor már az általános módon mégsem lesz redukálható a gráf, hanem egy speciális esettel kerülök szembe. A redukció következő eredménye már a Gömböc lenne és mivel a Gömböcnek nincs nyeregpontja, ezért az expanzió is speciális, ebből következő, hogy a Gömböcre redukálás is speciális módon történik. Viszont ezt az utolsó lépést már nem is kell elvégzem, hiszen ilyen helyzetben csak a Gömböc lehet a keresett topológiai ős.

4.2. Az algoritmus lépésszáma

A bemeneti fájl a gráf szomszédossági mátrixát tartalmazza. Jelöljük a gráf pontjainak számát n-nel. A mátrix mérete $(|S| + |U|) * |H|$, továbbá $n = |S| + |U| + |H|$, valamint tudjuk, hogy $|S| + |U| = |H| + 2$. Ezekből következik, hogy szomszédossági gráf mérete közelítőleg $\frac{n^2}{4}$. A beolvasást követően elvégezett műveletek:

- A stabil és instabil pontok számának meghatározása. A bemenetben nincs megadva a konkrét számok, a szomszédossági mátrix értékeinek előjeléből lehet következtetni az értékére. Ehhez a szomszédossági mátrixban meg kell keresnem a leginkább jobbra lévő pozitív értéket és a leginkább balra lévő negatív értéket. Ennek a lépésszáma: $O(n^2)$.
- A párhuzamos élek felderítése. Előfordul, hogy egy nyeregpont egy ponttal kétszeresen is össze van kötve. Ez a bemenetben úgy van jelezve, hogy a szomszédossági mátrix adott sorában csak három (vagy ha csak kettő) nem nulla elem van, és ezeket meg kell keresni a mátrixban. Ehhez szükséges a mátrix elemein végigmenni, így a lépésszáma: $O(n^2)$.
- A stabil pontok és az instabil pontok között futó élek meghatározása. Ez a bemenet során nincs megadva, nekem kell kiszámolni. Ehhez minden nyeregponton végig kell mennem és elvégezni a korábban leírt eljárást (a síkgráf minden tartományában a stabil pont és az instabil pont közé egy új él behúzását), ami, mivel minden nyeregpontnak négy fokszáma, konstans lépésben elvégezhető, de a mátrixot így is be kell járni. Összegezve ennek a lépésnek is a lépésszáma: $O(n^2)$.
- A beolvasás befejezéséhez már csak a fokszám megszámolása van hátra. Ehhez a két szomszédossági mátrixot kell egyszer-egyszer végignézni $\left(O(n^2)\right)$. Tehát összesen a beolvasás lépésszáma $O(n^2) + O(n^2) + O(n^2) + O(n^2) = O(n^2)$.
- Egy redukció végrehajtásához szükséges a fokszámot tároló tömbben egy másod- vagy negyedfokú csúcs kiválasztása. A tömb mérete $|S| + |U| = O(n)$. A nyeregpontok kiválasztásához, az előbb meghatározott pont adja meg, hogy a szomszédossági mátrix mely sorát kell átvizsgálni. Ha nyeregpontok megvannak,

akkor már csak a szomszédjaikat kell összevetni a meglévő stabil pontunk szomszédjaival, hogy megtaláljuk az összes pontot. Ehhez elegendő a szomszédossági mátrix egy során vagy oszlopán végigmenni. Így egy redukció lépésszáma $O(n)$. Legrosszabb esetben egy n pontú gráfból négy pontú lesz a végeredmény és mivel lépésenként két csúcsot veszünk el, ebben az esetben $\frac{n-4}{2} = O(n)$. Így összesen a redukció lépésszáma $O(n) * O(n) = O(n^2)$.

Így összességében a megvalósítás komplexitása $O(n^2)$. Felmerül a kérdés, hogy vajon lehetséges-e ezt a komplexitást csökkenteni. Kiderült, hogy igen és a következő fejezetben megmutatom, hogy hogyan.

5. A megvalósítás lépésszámának csökkentése

A transzformáció lépései az algoritmussal együtt adottak, viszont megvalósítani többféleképpen is lehetséges. Azonban a megoldások eltérő hatékonysággal futnak. Egyik megoldás az, amit az előző fejezetben bemutattam. Abban a megoldásban minden redukció során a szomszédossági mátrixban kellett keresnem a transzformációhoz szükséges csúcsokat, ami minden esetben egy sor vagy oszlop végignézését foglalta magában. Felmerült a kérdés, hogy az algoritmus megvalósítható-e kevesebb lépésszámmal is?

5.1. Megfelelő adatszerkezetek megválasztása

A Kolumbusz-algoritmusban által megadott redukciók során, az összes esetben a gráf csúcsainak száma kettővel változik. Ez azt jelenti, hogy az elvégzett redukciók száma a bemeneti gráf csúcsainak számától függ. Így a kérdés egy redukció komplexitásának csökkentésére fókuszálom. Egy redukcióra viszont már igaz, hogy lehetséges hatékonyabb implementációt készíteni az előző fejezetben ismertettetnél. Ehhez megfelelőbb adatszerkezeteket kell választanom a gráf tárolására valamint kiegészítőileg kell adatokat tárolnom a keresések gyorsítása érdekében.

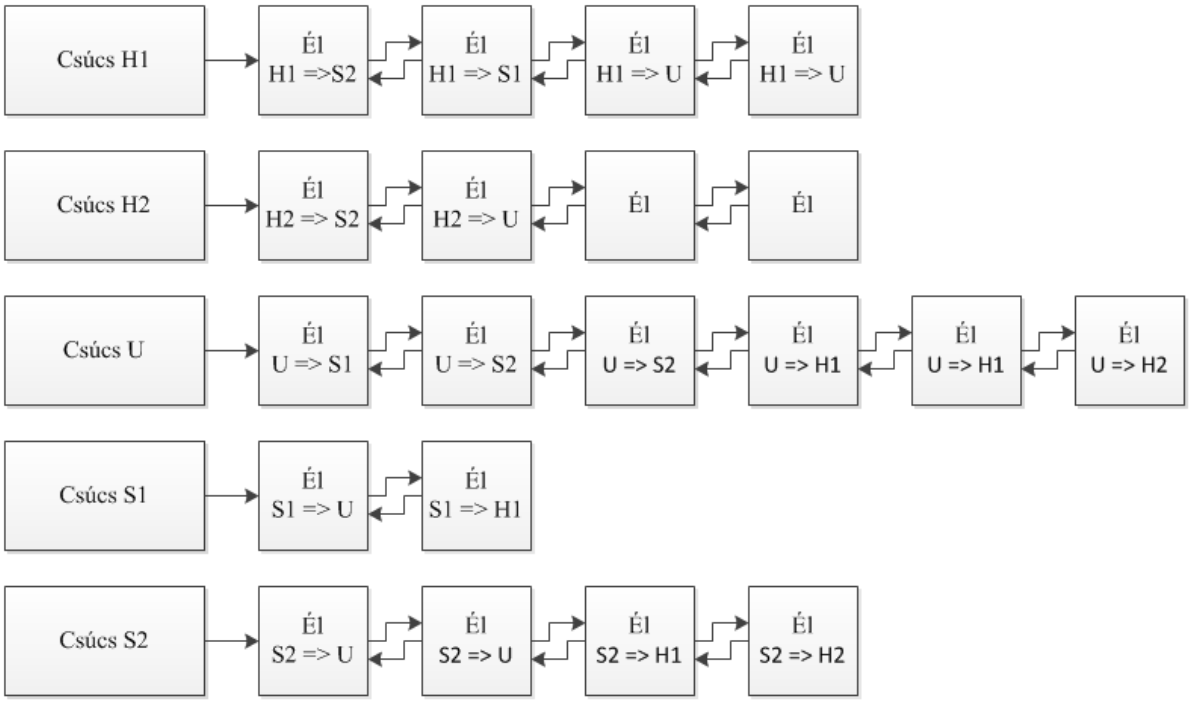
Az előző fejezetben kiszámoltam a megvalósított eset hatékonyságát az egyes lépésekre külön-külön. Egy redukciót nézve (mindkét esetben) a lépésszámra az $O(n)$ eredmény jött ki. Ennek a javítására dolgoztam ki elméletben egy hatékonyabb megoldást, más adatszerkezeteket alkalmazva. Beláttam, hogy egy redukció elvégezhető konstans lépésben is, így az ős megtalálásának komplexitása már nem négyzetes függvénye a bemenet méretének, hanem lineáris. Ehhez először bemutatom a felhasznált adatszerkezeteket és azt követően belátom, hogy ténylegesen konstans lépésben véghezvihető egy redukció.

Tekintsük át, hogy melyek azok a lépések, amiket konstans időben kell tudnunk elvégezni a redukció típusától függően.

1) Először nézzük egy negyedfokú csúcs redukciójának esetét. Az első lépés mindkét esetben egy csúcs kiválasztása, amely körül majd a redukció kivitelezve lesz. Ennek mind a négy szomszédját meg kell tudnunk találni. Ezt követően már csak egy pont hiányzik, amit viszont megtalálhatunk az egyik nyeregpont szomszédjai között. A hátralévő két lépés a megadott két csúcs törlése és velük együtt az össze hozzájuk kapcsolódó él törlése, valamint egy új él behúzása.
2) Abban az esetben ha másodfokú csúcsot redukálunk a helyzet eléggé hasonló. Találnunk kell egy másodfokú pontot, amelyik lehet akármelyik az összes másodfokú közül. Vennünk kell a szomszédjait. Ezek között van egy nyeregpont és egy stabil vagy instabil pont, a talált csúcstól függően. A nyeregpont szomszédjai közül így már csak egyet nem ismerünk, ami szintén szükséges a redukcióhoz.

Az első lépés megvalósításához megfelelő adatszerkezet két oda-vissza láncolt lista. Az egyikben az összes előforduló másodfokú, a másikban pedig a negyedfokú pontokat tárolom. Ezeket a listákat elneveztem másodfokú listának, valamint negyedfokú listának. Természetesen nyeregpontot nem tárolok, mivel egy nyeregpontból kiindulva nem lehetséges redukciót végrehajtani. Ha egy redukcióhoz szükségem van egy új, mondjuk, egy másodfokú csúcsra akkor mindössze annyit kell tennem, hogy leveszem a másodfokú lista elejéről az első élet és a lista második elemét beállítanom, mint a lista kezdő eleme. Ez a lépés független a bemeneti gráf méretétől. A következő lépés, hogy egy nyeregpont szomszédjait elérjem konstans lépéssel. Ez megoldható, ha a gráfot éllistás alakban tárolom. Mivel tudjuk, hogy minden nyeregpontnak minden esetben a fokszáma négy, így a nyeregpont ismeretében négy lépéssel megvalósítható. Így a pontok megtalálása végrehajtható konstans lépéssel.

Ezt követően a törlést kell konstans időben végrehajtanom. Ebben a lépésben éleket is törölnöm kell az éllistás adatszerkezetből. Ehhez nem elegendő tudnom, hogy melyik csúcs melyik élét akarom törölni, de tudnom kell, hogy az adott él, hol helyezkedik el a listában, különben végig kellene mennem rajta, ami viszont már nem lenne konstans lépésben megoldható, hiszen a lista mérete függ a bemenet méretétől. Ennek a megértéséhez tekintsünk az alábbi ábrára.



14. ábra Az éllista. A 13. ábra C_2 expanziója alapján készült

Ezen az ábrán az éllista található. Az ábrán a nyilak jelentése, hogy az adott élhez tárolunk egy mutatót, a másik élre. A csúcsokat egy tömbben tárolom, és minden csúcshoz kapcsolódik egy előre és hátra láncolt lista az élek tárolásához. Mivel a csúcsok száma dinamikusan nem nő a redukciók során, ezért nem probléma, ha egy fix méretű tömböm van. Az éllistában minden élet kétféle irányítással tárolok, mivel mindkét hozzá tartozó csúcs esetén ismernem kell az adott élet. Ahhoz, hogy egy élet törölni tudjak a gráfból, az éllistában

törölnöm kell az él mindkét irányítását. Mivel csak az egyik csúcs irányából ismerem csak az élet, ezért az éleknek tudniuk kell, hogy az ellentettjük hol vagy, ezáltal nem kell keresni a másik csúcs listájában. Itt nem csak azt tárolom, hogy az ellentett él melyik csúcsból indul, hanem azt is, hogy ez az ellentett él, kiinduló csúcsának listáján belül hol helyezkedik el.

Továbbá fontos, hogy az adatszerkezetből ki tudjam deríteni, hogy egy adott pont benne van-e a másodfokú listában vagy a negyedfokú listában és, ha igen akkor pontosan hol van.

A fentebb leírt adatszerkezetet a következő struktúrákkal valósíthatók meg C nyelven:

```
struct VERTEX {
        INTEGER idx;            // A csúcs sorszáma
        EDGE edge;              // Az egyik él, a lista eleje
        TYPE type;              // A csúcs típusa
        INTEGER degree;         // A csúcs fokszáma
        VERTEX vertex4;         // A csúcs helye a negyedfokú listában
        VERTEX vertex2;         // A csúcs helye a másodfokú listában
}
```

A struktúrában szereplő EDGE és VERTEX típusú változók mutatók. A vertex4 és a vertex2 változókban tárolom, hogy a csúcs benne van e a másodfokú listában vagy a negyedfokú listában. Így az listán belül is konstans lépésben megtalálható. A másik struktúra az él struktúrája, szintén C nyelven megvalósítva:

```
struct EDGE {
        EDGE prev               // A listában a megelőző él
        EDGE next               // A listában a következő él
        EDGE pair               // A éllistában az él párja, minden élnek van párja
        VERTEX dest             // A él végén lévő csúcs
}
```

Ez a struktúra egy előre-hátra láncolt lista eleme, így tárolom benne az élet megelőző és az azt követő élet, amelyek ugyanabból a csúcsból indulnak ki, valamint az él ellentettjét, amelyik visszafelé mutat. Valamint tárolom, hogy az él végpontján melyik csúcs van.

5.2. A megvalósítás menete

Az redukcióhoz használt adatszerkezet bemutatását követően, megmutatom, hogy lehetséges egy redukciót konstans lépésben elvégezni. Ehhez vegyük sorra az elvégezendő műveleteket a két esetre.

5.2.1. Negyedfokú csúcs esetén

A szemléltetés során a csúcsok neveit, a 13. ábrán megadottak szerint használom. A redukció elvégzéshez egy negyedfokú csúcsra van szükség. Ilyet egy lépésben találok, ezért vezettem be a negyedfokú listát. A negyedfokú lista bármely eleme megfelelő, de az elsőt választom. Élek a feltételezéssel, hogy ez a lista elején lévő csúcs stabil volt, mivel ha instabil lenne a redukció ugyanúgy végrehajtható, csak a stabil és instabil pontok mindenhol felcserélődnek. A lista első elemét egy lépésben megkapjuk, egy másik lépés a lista első elemének átírása a korábbi másodikra. Ez a csúcs lesz S1.

A következő feladat H1, H2, U1 és U2 megtalálása. Ezek a csúcsok mind S1 szomszédjai, így az éllistában kell végignéznem a S1-hez tartozó láncolt listát. A listában pontosan négy elem van, és mindegyik elem mutat a keresett négy csúcs egyikére. Így legfeljebb nyolc lépéssel megtalálhatom a keresett csúcsokat. Már csak S2 megtalálása van hátra. A legkézenfekvőbb lépés H2 szomszédjai között keresni, hiszen H2 nyeregpont így négy szomszédja van. Hasonlóan, mint az előbb, amikor a másik négy (H1, H2, U1 és U2) pontot kerestem, S2 is megtalálható legfeljebb nyolc lépésben.

A redukció végrehajtásához törölnöm kell az következő éleket: S1=>H1, S1=>U1, S1=>U2, S1=>H2, H2=>U1, H2=>U2, H2=>S2. Röviden S1 és H2 összes élét törölnöm kell. Ez összesen hét él törlése, mivel mindkét csúcs negyedfokú és össze vannak kötve. Felhasználom, hogy minden él ismeri az ellentettjét, és hogy egy előre-hátra láncolt listában ismerem a törlendő elem helyét, akkor két lépéssel ki tudom törölni (két mutató átállítása). Így mind a hét élet ki tudom törölni, legfeljebb tizennégy lépésben.

A redukció befejezéséhez be kell húznom egy új élet S2 és H1 közé. Ehhez a két csúcshoz tartozó éllista elejére kell beszúrnom az élet és az ellentettjét. A redukció során módosítottam U1 és U2 fokszámát így lehet, hogy bekerülnek a másodfokú vagy negyedfokú listába. Mivel abban a láncolt listában is ismerem a helyüket, így onnan kivenni vagy betenni is elvégezhető konstans lépésben. Mivel a redukció minden részére megadható felső, konstans lépésszám korlát, ezért egy redukció is elvégezhető konstans lépés végrehajtásával. (Részletesebben lásd. Függelék [2].)

5.2.2. Másodfokú csúcs esetén

Másodfokú csúcs redukálása esetén, az eljárás menete nagyon hasonló a negyedfokú csúcs esetéhez. A különbség a mintához (13. ábra) való illesztésben van.

6. Tesztelés

Miután elkészítettem az implementációt, kaptam bemeneti gráfokat, amelyekre meg kellett határoznom a topológia őst. A kapott gráfok első része 3D szkennerrel beolvasott természetes kavicsok, a másik része generált gráfok. A második esetben a generált gráfokra meghatározták [2], hogy a gráfok közül hány topológiai ős található. A tesztelés során az implementációm, gyakorlatban is ugyanazt az eredményt szolgáltatta.

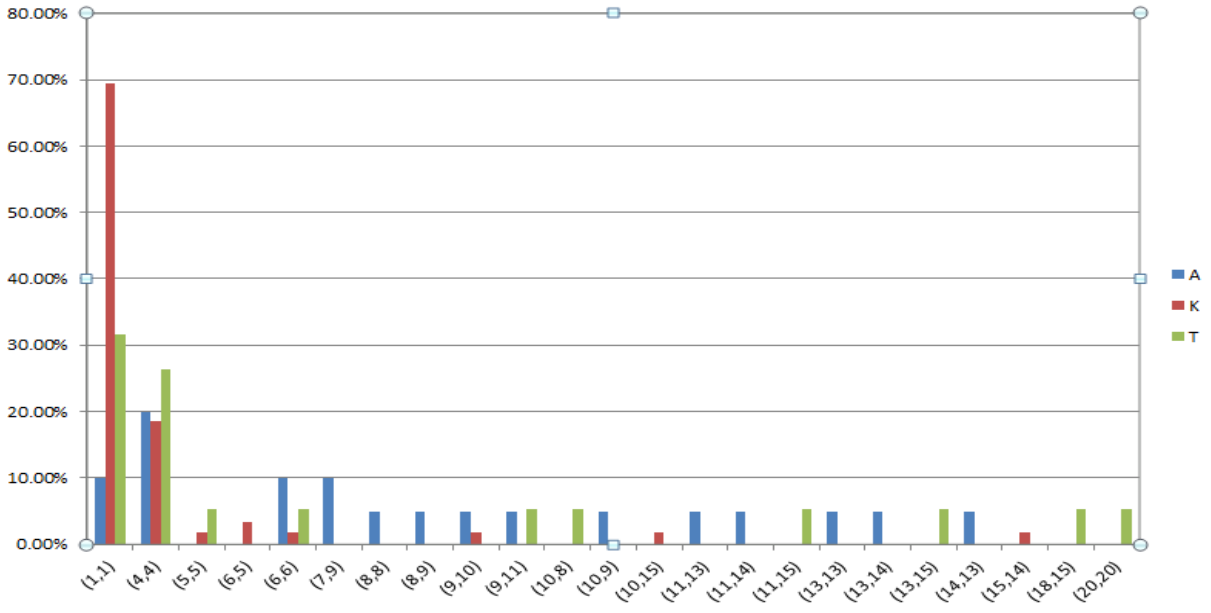
6.1. Teszt gráfok ismertetése

A teszt gráfok első része a Szilárdságtani és Tartószerkezeti Tanszék munkatársai által 3D scannerrel beolvasott 98 természetes kavics gráfjai. Ezek a kavicsok geológiai szempont A, K illetve T osztályokba sorolhatóak [7]. A tesztelés során a topológia ősök az eltérő osztályok között eltérő gyakorisággal fordultak elő.

Rendelkezésemre állt továbbá az összes lehetséges gráf, generálva $S + U \leq 10$ méretkorlátig. Ahhoz, hogy ezekeknek a gráfoknak is meg tudjam határozni az ősét, szükségem volt a beolvasás módosítására, mivel ezek a gráfok szomszédossági mátrix helyett, éllistával voltak megadva. A program további részein nem kellet változtatnom.

6.2. Eredmények, statisztikák

A természetes kavicsok beolvasás során három osztályba (A, K és T osztályokba) lettek sorolva. A különböző osztályok esetén az előforduló ősök aránya különböző. Ezeket az előfordulási arányokat az alábbi diagram mutatja.



15. ábra A topológia ősök gyakorisága az egyes osztályok esetében

Ezen a diagramon látható, hogy az A, K és T osztályú kavicsok között, a meghatározott topológiai ősök, melyik elsődleges osztályba tartoznak (Függelék [1]).

A generált gráfok esetében, a topológia ős keresését több mint hetven-ezer gráfra futtattam le. A kevesebb mint tizennégy pontú gráfok esetén mindössze a Gömböc és a tetraéder fordult elő, mint topológiai ős. Például a tizenhat pontú gráfok esetén a 9172 tesztesetből 9165 esetben a Gömböcöt kaptam mint topológiai ős, négy esetben a tetraédert és mindössze a három tesztesetben kaptam magasabb pontú gráfot, mint topológiai ős.

7. Összegzés

A feladat során implementáltam a topológia ős keresésére szolgáló algoritmust. Az implementálást először egy kevésbé hatékony eljárással végeztem. Felvetődött a kérdés, hogy lehet-e kevesebb komplexitással is megvalósítani. A válasz igen. Kidolgoztam olyan módszert amivel a topológiai ős keresése lineáris lépésben történik a bemenet függvényében. A kész implementációt több mint 70 ezer gráf esetén futtattam le. Az eredmény a legtöbb esetre a Gömböc lett a topológia ős, viszont nagyobb pontszámú gráfok esetén, már új, kevésbé gyakori topológiai ősök is előfordultak.

Irodalomjegyzék

[1] G. Domokos and P. Várkonyi. Static equilibria of rigid bodies: dice, pebbles and the Poincaré–Hopf theorem. Journal of Nonlinear Science, 16:255–281, 2006.

[2] Kápolnai Richárd, Domokos Gábor és Szabó Tímea: Másodlagos egyensúlyi osztályok gráfelméleti származtatása, XI. Magyar Mechanikai Konferencia, 2011, Miskolc

[3] Sipos András Árpád, Szabó Tímea és Domokos Gábor. Egyensúlyok azonosítása kavicsokon és a falkákat felismerő algoritmus.. In XI. Magyar Mechanikai Konferencia, 2011, Miskolc.

[4] H. Edelsbrunner, J. Harer, and A. Zomorodian, *Hierarchical morse complexes for piecewise linear 2-manifolds*, in Proceedings of the seventeenth annual symposium on Computational geometry, SCG ’01, New York, 2001, ACM, pp. 70–79.

[5] Katona Gyula Y., Recski András, Szabó Csaba (2002) "A számítástudomány alapjai", Typotex Kiadó, Budapest

[6] G. Brinkmann and B. D. McKay, *Fast generation of planar graphs*, MATCH Communicationsin Mathematical and in Computer Chemistry, (2007) **58**, pp. 323–357.

[7] Szabó Tímea, Sipos András Árpád és Domokos Gábor. Kavicspopulációk és egyensúlyok statisztika. In XI. Magyar Mechanikai Konferencia, 2011, Miskolc.

Tartalomjegyzék

Függelék

[1] **Táblázat a 15. ábra diagramjához.**

A	K	T
10.00%	69.49%	31.58%
20.00%	18.64%	26.32%
0.00%	1.69%	5.26%
0.00%	3.39%	0.00%
10.00%	1.69%	5.26%
10.00%	0.00%	0.00%
5.00%	0.00%	0.00%
5.00%	0.00%	0.00%
5.00%	1.69%	0.00%
5.00%	0.00%	5.26%
0.00%	0.00%	5.26%
5.00%	0.00%	0.00%
0.00%	1.69%	0.00%
5.00%	0.00%	0.00%
5.00%	0.00%	0.00%
0.00%	0.00%	5.26%
5.00%	0.00%	0.00%
5.00%	0.00%	0.00%
0.00%	0.00%	5.26%
5.00%	0.00%	0.00%
0.00%	1.69%	0.00%
0.00%	0.00%	5.26%
0.00%	0.00%	5.26%

[2] **Egy negyedfokú csúcs redukciója konstans lépésben**

Kell egy negyedfokú nem nyeregpont a redukció elvégzéséhez. Már beláttuk, ha találunk egy ilyet, akkor biztosan elvégezhető a redukció, tehát minden a redukcióhoz szükséges csúcsot, illetve élet megtalálunk, ott ahol lennie kell. A negyedfokú csúcsokhoz van egy külön listánk, amiben mindig frissítjük a csúcsokat, így biztos, hogy a benn lévő összes csúcs megfelel a kritériumoknak. Ezek szerint a lista bármelyik elemét kiválaszthatjuk, az egyszerűség kedvéért válasszuk az első elemét. Ez az elem legyen S1. Ezt követően állítsuk be a második elemet első elemnek, ehhez mindössze egy mutatót kell átírnunk, hogy ne az eddig első elemre mutasson, hanem a másodikra.

Ezt követően szükségünk van a többi öt pontra. Ennek az öt megtalálásához szükségünk van S1 szomszédjaira (H1, H2, U1, U2), amely négy ezen öt pont közül való. A vertex struktúrában tároljuk a csúcs indexét, aminek segítségével egy lépésben megtalálhatjuk a csúcsot az éllistában, és az élei olvasása további négy lépést vesz igény, majd az élek mutatói

alapján, élenként még egy lépés a szomszédok megtalálása. Így ennek a részfeladatnak a végrehajtásához összesen kilenc lépés szükséges.

Ezt követően meg kell találnunk S2-t is. Ehhez H2 szomszédjai között kell keresni. Ha egy lépésnek tekintünk egy összehasonlítást és egy struktúra elérését is egy mutatón keresztül, akkor ehhez legrosszabb esetben (ha a listában utolsó helyen áll a keresett csúcs) kell négy él elérése és minden él esetében meg kell vizsgálni, hogy a benne található dest nevű mutató, egy már megtalált csúcsra mutat-e vagy sem. Ha eljutunk az utolsó élhez is akkor már biztosan megtaláltuk, ezért nem szükséges az összehasonlítás elvégzése. Ehhez legfeljebb $4 + 3 * 3 = 13$ lépés szükséges.

Ezt követően két csúcsot kell törölnünk. Az éllista tömbjéből nem muszáj törölnünk, elegendő, ha csak minden hivatkozást törlünk az adott csúcsra. Mindkét csúcs negyedfokú, szóval mindkettő törléséhez ugyanannyi lépés kell majd. Egy él törléséhez először szükséges a törölni a párját. Ezért volt fontos, hogy tároljuk a pár helyzetét, hiszen ily módon nem kell keresni az élet, csak egy oda-vissza láncolt listában kell megoldani a törlését. Ehhez először a megelőző él next változóját kell átállítani a törlendő él next változójára, majd a következő él prev változójának kell megadni, a törlendő él prev változójának értékét. Ennek a megvalósítása során két lépést kell végrehajtani, a két mutató átállítását. Ezt csúcsonként négyszer kell elvégezni és összesen van két csúcs, tehát tizenhat lépés. A redukció befejezéséig hátralévő lépés S2 és H1 összekötése egy új éllel. Ehhez az éllistában mindkét csúcshoz hozzá kell adnunk egy-egy új élt. Az új él elhelyezése bárhol történhet az éllistában, viszont a láncolt listának csak az elejét ismerjük, így konstans lépésben csak az elejére tudjuk beszúrni. Ezzel a lépésben nem rontjuk el a láncolt lista szerkezetét, ezért tegyük is ide az élet és az ellentettjét.

A redukció során módosítottuk néhány csúcs fokszámát, U1 és U2 fokszáma kettővel csökkent. Két esetben kell további módosításokat végrehajtanunk. Ha valamelyik fokszáma hat volt és most négy lett, akkor a negyedfokú csúcsokat tároló listába kell beszúrni. Ez könnyen megoldható, csak a lista végére kell szúrni, valamint az éllistában meg kell adni a csúcs vertex4 változójának, a negyedfokú csúcsokat tároló listabeli helyének címét. Ez két értékadással megoldható. A másik eset, amikor U1 vagy U2 negyedfokú volt csúcs volt. Ilyenkor először az adott csúcsot a negyedfokú csúcsokat tároló listából kell törölni, majd hozzáfűzni a másodfokú csúcsokat tároló listához, majd a struktúra vertex4 és vertex2 nevű változóit megfelelően változtatni. Az első lépéshez segítséget nyújt az előbbi változó, aminek segítségével két lépésben ki lehet fűzni a csúcsot a láncolt listában, majd ezt követően a másodfokú csúcsokat tároló lista végére kell tenni, ami egy lépésben megvalósítható és további egy lépés az, hogy beállítjuk a csúcsban a vertex2 változót. Végeredményként elértem, hogy egy negyedfokú csúcs redukciójához minden része konstans lépésszámmal megvalósítható.