Eötvös Loránd Tudományegyetem
Természettudományi Kar

A Pebble game és kapcsolódó algoritmusok implementálása

Önálló projekt
beszámoló

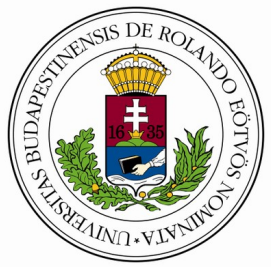
Matúz Lóránt

Alkalmazott Mathematika MSc

Témavezető:

Madarasi Péter

Operációkutatási Tanszék



Budapest

2022

1 A téma és motivációja

A projekt keretében a (k, l)-ritka gráfok felismerésére szolgáló Pebble Game algoritmusokkal [1] és azok implementálásával foglalkozunk. Egy multigráfot (k, l)-ritkának nevezünk, ha bármely X csúcsrészhalmaza által feszített élek száma legfeljebb $\max\{k|X| - l, 0\}$. A témakör fontos alkalmazásai közé tartozik például a Nash-Williams tételből [2] ismert éldiszjunkt feszítőfákra való felbontás és Tutte tételből [3] az éldiszjunkt erdőkkel vett fedés. Ezentúl megemlíthető még a merevség is, mely síkban Laman tétele [4] szerint ekvivalens a $(2, 3)$-ritkasággal, három dimenzióban pedig egy szükséges feltételt jelent a $(3, 6)$-ritkaság [5].

2 Az előzetes munkák

A BSc szakdolgozatom témája is a (k, l)-ritkaság és a Pebble Game algoritmusokhoz kapcsolódott, így a félév kezdetén már széleskörű előzetes ismeretekkel rendelkeztünk. Alaposan feldolgoztuk a rendelkezésre álló forrásokat mind a Pebble Game algoritmusokkal [1, 6], mind a hozzájuk fűződő alkalmazásokkal [2, 3, 7] kapcsolatban. Az alábbiakban foglaljuk össze a korábbi munkánk főbb pontjait:

- A ritka gráfok tulajdonságainak elemzése, a komponensek (legnagyobb ritkák) karakterizációja. A ritka gráfok élhalmazán definiált matroid struktúra megértése.
- A Pebble Game algoritmusok különféle fajtáinak feldolgozása és implementációja a LEMON [8] könyvtárban C++ nyelven.
- A Component Pebble Game algoritmus implementációjának újragondolása. Egy másik megközelítés bevezetése: az implementációban a korábbi leírással [6] ellentétben a komponensek tárolásában a csúcsok játszanak szerepet, nem az élek.
- A (k, l)-ritkaság általánosítása, azaz a (j, k, l)-ritakság bevezetése, mely egy j alsó korlátot követel a sértő csúcsrészhalmazok méretére. Az NP-nehézség igazolása (j, k, l)-ritka gráfok felismerésére vonatkozóan.
- A (j, k, l)-ritkaságra tekintett $l = 2k$ eset korábbi algoritmusának egy nagyságrendű javítása, és ennek implementálása.
- A Pebble Game alkalmazásainak implementálása: éldiszjunk feszítőfákra bontás, és erdőkkel vett fedés, valamint a tényleges feszítőfák és erdők megtalálásához szükséges fenyők és fenyvesek keresésének algoritmusai [9, 904-928. o.].
- Az implementáció tesztelése, futási idők mérése, algoritmusok összevetése.

3 Az e félévi munkák

A szakdolgozat folytatásaként a projekt számos további lehetőséget tartogatott. Így tehát az implementáció további bővítéséve, javításával és hatékonyabbá tételével foglalkoztunk. Részleteit tekintve a következőképp foglalhatjuk össze mindezt:

- **Duálisok:** A feszítőfák, erdők, fenyők és fenyvesek kereséséhez lehetőséget biztosítottunk a duális megoldás kiolvasására az implementációinkban.

- **BFS:** Hatékonyabb implementációt adtunk meg a BFS algoritmusra, mely a Pebble Game algoritmusok mindegyikének alapját képezte. Többek közt kevesebb lépésben inicializáljuk (újra) a BFS objektumokban használt sort és hatékonyabb útkereső függvényeket adtunk meg.

- **KINARI:** Az implementációinkat összevetettük a korábbról már ismert, proteinek merevségét vizsgáló KINARI projekttel [10, 11], mely szintén a Component Pebble Game algoritmusát implementálta, azonban nem nyitott forráskódú. A teszteket elvégeztük többféle gráftípuson a merevség szempontjából: random, síkbeli merev, legnagyobb ritka, molekula és protein gráfokon. Arra a következtetésre jutottunk a tesztek alapján, hogy a Pebble Game algoritmusok LEMON-beli implementációi egy nagyságrenddel hatékonyabbnak bizonyulnak majdnem minden gráftípuson.

- **Heurisztikus élsorrend:** A Basic Pebble Game algoritmusában az élek feldolgozásának sorrendje szabadon megválasztható, ennek heurisztikus választásával is foglalkoztunk. Észrevettük annak az egyszerű ténynek a kardinális jelentőségét, hogy amennyiben megtaláltuk a maximális ritka részgráfot, az algoritmus terminálhat. Ezért tehát olyan heurisztikákat igyekeztünk választani, melyek nagy valószínűséggel elfogadott éleket preferálnak. Ezek alapján tehát az alábbi éleket preferáló heurisztikákat mértünk össze:

  - **Alap**: implementáció alapvető éleinek sorrendje.
  - **Minfok**: minimális fokszámú csúcsra illeszkedő élek sorrendje.
  - **Minill**: legkevesebb rá illeszkedő élt választó élek sorrendje.
  - **Maxkifok**: végpontjain maximális kifokösszegű élek sorrendje.

  Ezen heurisztikákat a fentebb már említett gráftípusok mindegyikén teszteltük. Összefoglalva azt mutatták az eredmények, hogy a futási idők hatalmas mértékben függhetnek az alkalmas élsorrend megválasztásától. Továbbá azt láthattuk, hogy a maxkifok heurisztika egyértelműen jobbnak bizonyult a többinél, valamint hogy a minfok is érdemben javított a futási időn és ezen élsorrend kiszámítása könnyen megtehető. Megfigyeltünk néhány gráftípuson fluktuációt is a futási időkben.

4 Eddigi publikációk és tervek

Az előzőekben röviden bemutatott eredményekből egy TDK dolgozat készült a félév során. Ebben bemutattuk a korábbi eredményeinket is, többek közt az általánosítás NP-nehézségét és az $l = 2k$ esetére vonatkozó javított algoritmust. Emellett a dolgozat tartalmazta még az implementációk összehasonlítását a KINARI projekttel, valamint a heurisztikák összehasonlítását.

Az elkészített implementációk terveink szerint hamarosan be is kerülhetnek a LEMON könyvtárba, ezzel a merevség tesztelés mellett például éldiszjunkt feszítőfák keresésére lehetőség nyílik a LEMON segítségével. Így tehát az általunk ismert első nyílt forráskódú projekt, mely ezen funkciókat is magába foglalja.

A LEMON könyvtár bővítése mellett egy tech reportot is fogunk készíteni, mely bemutatja az implementáció részleteit.

Hivatkozások

[1] A. Lee and I. Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics 308*, 2008.

[2] C. S. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, 1961.

[3] W. T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal of the London Mathematical Society*, 1961.

[4] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 1970.

[5] N. Katoh. A proof of the molecular conjecture. *Discrete Comput Geom 45*, 2009.

[6] A. Lee, I. Streinu, and L. Theran. Finding and maintaining rigid components. *Canadian Conference on Computational Geometry*, 2005.

[7] B. Servatius J. Graver and H. Servatius. Combinatorical rigidity. *American Mathematical Society*, 1993.

[8] D. Balázs, A. Jüttner, and P. Kovács. LEMON - an Open Source C++ Graph Template Library. *Electron. Notes Theor. Comput. Sci.*, 2011.

[9] A. Schrijver. Combinatorial optimization. *Springer*, 2004.

[10] KINARI: Kinematics and Rigidity. `http://kinari.cs.umass.edu`.

[11] N. Fox, F. Jagodzinski, Y. Li, and I. Streinu. KINARI-web: a server for protein rigidity analysis. *Nucleic Acids Research*, 2011.