Specifikace softwarového díla
&
Časový plán implementace
pro

Generování jednoduchých nakreslení grafů

Filip Čermák

6. června 2020

Verze 1.

Anotace

Cílem projektu je generovat jednoduchá nakreslení úplných a případně obecných grafů, na kterých poté půjde ověřovat platnost známých kombinatorických domněnek. Jako hlavní motivaci máme počítání průsečíkových čísel úplných grafů.

Obsah

Tabulka revizí

Jméno	Datum	Důvod změny	Verze
Filip Čermák	30.4.2020	Finální verze specifikace	1.0

1 Základní informace

1.1 Popis a zaměření softwarového díla

Jednoduchá nakreslení grafů jsou ta, kde se žádné dvě hrany nekříží více než jednou, přičemž průnik ve vrcholu se počítá také jako křížení.

Program poslouží ke generování jednoduchých nakreslení úplných, případně obecných grafů. Na těchto vygenerovaných nakresleních půjde poté ověřovat platnost známých domněnek, například o průsečíkových číslech.

Dále *průsečíkové* číslo grafu je minimální počet křížení křivek reprezentujících hrany přes všechna nakreslení daného grafu.

Jednou z motivací je zkoumání struktury jednoduchých nakreslení grafů pomocí tzv. *k-hran*, což by mohlo dát nový náhled například u Harraryho–Hillovy domněnky o průsečíkových číslech úplných grafů.

1.2 Použité technologie

- Programovací jazyk C++, std/c++17
- Programovací jazyk C#, Visual Studio 2019 Community
- C# WindowsForms (knihovna tříd pro grafické rozhraní)
- Graph# (knihovna pro vykreslování grafů, `https://archive.codeplex.com/?p=graphsharp`)

1.3 Odkazy(Reference)

[1] Martin Balko, Radoslav Fulek a Jan Kynčl. „Crossing numbers and combinatorial characterization of monotone drawings of K_n“. In: *Discrete Comput. Geom.* 53.1 (2015), s. 107–143. ISSN: 0179-5376. DOI: 10.1007/s00454-014-9644-z. URL: https://doi-org.ezproxy.is.cuni.cz/10.1007/s00454-014-9644-z.

[2] Lowell Beineke a Robin Wilson. „The early history of the brick factory problem“. In: *Math. Intelligencer* 32.2 (2010), s. 41–48. ISSN: 0343-6993. DOI: 10.1007/s00283-009-9120-4. URL: https://doi-org.ezproxy.is.cuni.cz/10.1007/s00283-009-9120-4.

[3] Jürgen Pammer. „Rotation systems and good drawings“. Dipl. Austria: Institute for Software Technology Graz University of Technology, 2014.

2 Stručný popis softwarového díla

2.1 Důvod vzniku softwarového díla a jeho základní části a cíle řešení

Cílem projektu je pokusit se vyvrátit či případně na malých případech ověřit známé domněnky o jednoduchých nakresleních grafů.

V současnosti je nám známá pouze jedna databáze jednoduchých nakreslení od Jürgena Pammera [3]. Tato databáze ovšem není volně dostupná a obsahuje pouze jednoduchá nakreslení úplných grafů. K vytvoření databáze jednoduchých nakreslení úplných grafů použijeme podobné metody a pokusíme se je rozšířit také na obecné třídy grafů.

Problém rozhodnout, jaký je minimální počet křížení v nakreslení daného grafu, je obecně velice obtížný. I pro speciální třídy grafů se jedná o jedny z nejznámějších a nejstarších kombinatorických problémů o grafových nakresleních. Například problém určit minimální počet průsečíků v nakresleních úplných bipartitních grafů $K_{m,n}$ je starý přes 70 let a je známý jako *Zarankiewiczova domněnka* [2]. Jako první se těmito problémy zabýval matematik Paul Turán, kterého podobné otázky napadly, když byl v roce 1944 nucen pracovat v továrně na cihly. Turán zde musel převážet cihly z několika továren do skladů pomocí vozíků, které pracovníci přesouvali po kolejnicích. Největším problémem při převozu byla křížení kolejnic, protože na nich často cihly z vozíků popadaly. Turána napadlo, že rozvržení kolejnic minimalizující počty těchto křížení by vedlo k úspoře času a práce. Nebylo ovšem jasné, jak by takové rozvržení mělo vypadat a pro obecný počet n továren a m skladů se dodnes jedná o nevyřešený problém. Podobná přes sto let stará úloha zvaná *Tři domy a tři studně* je vlastně problémem určení minimálního počtu průsečíků v nakresleních grafu $K_{3,3}$. Zde lze ovšem poměrně snadno nahlédnout, že jedno křížení je nutné a zároveň dostačující.

Určení minimálního počtu průsečíků v nakresleních úplného grafu K_n na n vrcholech je také těžkým problémem známým pod mnoha jmény [2], například jako *Hararyho–Hillova domněnka*. Tento velice známý a dosud nevyřešený problém pochází z 50. let, kdy jej poprvé formuloval umělec Anthony Hill. Na tomto problému během posledních desítek let pracovala spousta známých matematiků a přesto, že existuje vzorec pro domnělý minimální počet průsečíků v nakreslení K_n, nepodařilo se dodnes dokázat, že neexistují lepší nakreslení. Narozdíl od Zarankiewiczovy domněnky se však v posledních několika letech podařilo u Hararyho–Hillovy domněnky dosáhnout významného pokroku.

Cílem tohoto projektu by bylo s využitím vytvořené databáze jednoduchých nakreslení hlouběji prozkoumat metody, za pomoci kterých bylo těchto pokroků dosaženo, a případně dále prohloubit naše znalosti o nakresleních minimalizujících počty průsečíků úplných ale i obecných grafů.

2.2 Hlavní funkce

Program se bude skládat ze dvou částí. Z programu pro generování jednoduchých nakreslení (**generátor**) a z programu (**vizualizér**) pro vizualizaci nakreslení vystoupených generátorem.

Kód generátoru bude psaný v jazyce C++ a bude implementovat datovou strukturu **Half-edge**. Kód vizualizéru bude psaný v jazyce C#. Vizualizér poté vykreslí nakreslení vystoupené generátorem přehledně pro uživatele, který na něm bude moci ověřovat nějakou z domněnek o průsečíkových číslech, či jinou z domněnek o nakresleních grafů.

2.3 Motivační příklad užití

Příkladem problému, k jehož řešení by naše databáze mohla přispět, je slavná Harraryho–Hillova domněnka, která říká, že průsečíkové číslo úplného grafu K_n je přesně $Z(n) := \frac{1}{4}\left\lfloor\frac{n}{2}\right\rfloor\left\lfloor\frac{n-1}{2}\right\rfloor\left\lfloor\frac{n-2}{2}\right\rfloor\left\lfloor\frac{n-3}{2}\right\rfloor$. Jsou již známá nakreslení grafu K_n se $Z(n)$ průsečíky, ale ukázat, že vždy existuje alespoň $Z(n)$ průsečíků je otevřeným problémem téměř 70 let.

V článku [1] je uvedeno zobecnění Harrahyho–Hillovy doměnky, které využívá *k-hran*. Mějme D jednoduché nakreslení grafu K_n. Poté se podívejme na orientovaný oblouk γ reprezentující hranu $\{u, v\}$ pro různé vrcholy u a v. Pro libovolný jiný vrchol w řekneme, že je nalevo (napravo) od γ, pokud topologický trojúhelník uvw je orientovaný proti (po) směru hodinových ručiček. Hrana $\{u, v\}$ je *k-hranou*, pokud má právě k vrcholů napravo nebo nalevo.

Zadefinujme $E_k(D)$ jako počet k hran v daném nakreslení D. Položme

$$E_{\leq k}(D) = \sum_{i=0}^{k} E_i.$$

Obdobně můžeme zadefinovat

$$E_{\leq\leq k}(D) = \sum_{i=0}^{k} E_{\leq i}(D) = \sum_{i=0}^{k}(k-i+1)E_i(D)$$

a zcela analogicky

$$E_{\leq\leq\leq k}(D) = \sum_{i=0}^{k} E_{\leq\leq i}(D) = \sum_{i=0}^{k}(k-i+1)E_{\leq i}(D) = \sum_{i=0}^{k}\binom{k+2-i}{2}E_i(D).$$

Zobecněná Harryho–Hillova domněnka [1] říká, že pro každé jednoduché nakreslení D grafu K_n a každé k, kde $0 \leq k < \frac{n}{2} - 1$ platí $E_{\leq\leq\leq k}(D) \geq \binom{k+4}{4}$.

Existuje verze této domněnky pro obecné grafy, která říká, že pokud má graf alespoň $\binom{2k+3}{2}$ hran, pak je $E_{\leq\leq\leq k}(D) \geq \binom{k+4}{4}$. Tato domněnka je motivací pro generování jednoduchých nakreslení obecných grafů.

2.4 Prostředí aplikace

Aplikace bude primárně cílená na Windows, ve kterém bude laděna. Program generátoru by měl fungovat i na operačním systému Linux.

3 Vnější rozhraní

3.1 Uživatelské rozhraní, vstupy a výstupy

Oba programy budou lokální. Generátorem vystoupená jednoduchá nakreslení budou uložená do souboru, ve kterých budou dostupná pro program vizualizér.

Typická interakce mezi uživatelem a vizualizérem bude možnost vytvoření si svého nakreslení grafu, zobrazení konkretního nakreslení vystoupeného generátorem a provádění jeho úprav (přidání/odebírání vrcholů, hran). Následně si bude uživatel moci zobrazit různé informace o daném nakreslení, které budou záviset na tom, co chce zrovna zkoumat, například $E_k(D), E_{\leq k}(D), E_{\leq\leq k}(D)$ nebo $E_{\leq\leq\leq k}(D)$, či počty průsečíků.

3.2 Rozhraní se software

Databáze jednoduchých nakreslení vystoupených generátorem bude poté volně přístupná na `http://bitbucket.org/`, pokud by si chtěl někdo otestovat svou oblíbenou domněnku.

Nakreslení generátoru budou uložena ve formě fingerprintu (otisku), což je úsporný způsob reprezentace jednoduchého nakreslení grafu, použitý v [3]. Pro graf G a vrchol $v \in V(G)$ buď π_v cyklickou permutací hran sousedících s tímto vrcholem. *Rotačním systémem* je potom množina $\pi = \{\pi_v : v \in V(G)\}$ těchto cyklických permutací, nebo též *rotací*. *Fingerprint (otisk)* je kanonický rotační systém, kde vybereme lexikograficky minimální posloupnost rotací kolem vrcholů. Označme jako $\rho(v)$ lexikograficky minimální rotaci okolo vrcholu v. Potom otiskem je $\rho = (\rho_{min}(2), \ldots, \rho_{min}(n))$ ($\rho_{min}(1)$ BÚNO můžeme vynechat, protože v otisku platí $\rho_{min}(1) = (2, 3, \ldots, n)$).

4 Detailní popis funkcionality

4.1 Generování jednoduchých nakreslení K_n

Program generování probíhá na struktuře **Half-edge**. Pro graf K_n prvně vytvoříme „hvězdu“, která představuje středový vrchol 1 spojený se všemi ostatními. Poté se budeme pokoušet vytvořit nové hrany, které ještě zbývají. Tyto vrcholy nebudou vrcholy v běžném slova smyslu. Budou uchovány jako kružnice hran (tedy budou tvořit „jednostrannou“ stěnu). Každá taková kružnice bude obsahovat $n - 1$ vrcholů, které budou uspořádány dle rotace daného vrcholu. Rotační systém totiž již kombinatoricky určuje jednoduché nakreslení.

Budeme si uchovávat pole $blocked[a][b][c][d]$, které nám bude říkat, jestli se hrany $\{a, b\}$ a $\{c, d\}$ mohou protnout, přičemž hrany, které mají společný vrchol, již považujeme za protnuté, aby bylo nakreslení jednoduché. Dále budeme mít seznam segmentů $segments[s]$, což jsou již nakreslené části hran, a také pole $indices[a][b]$, které si pro hranu $\{a, b\}$ bude pamatovat z jakého segmentu vede. Poté už bude jednoduché zjistit, mezi kterými vrcholy na daných kružnicích hranu právě vytváříme.

Vždy, když zjistíme, že danou hranu vytvořit za daných podmínek neumíme, vynoříme se z rekurze a zkusíme další možnost.

4.2 Vizualizace jednoduchých nakreslení

Poté bychom chtěli tato nakreslení vizualizovat, abychom mohli vidět jeho strukturu. To bude zajišťovat naše aplikace vizualizéru v jazyce C#. Uživatel si bude moci přidat vrcholy. Nechat je spojit hranou, odebrat vrchol či hranu. Přidaní vrcholů znamená kliknutí na již stávající vrchol a protáhnutí hrany až do nového bodu v rovině, kde vytvoří se vytvoří bod (kvůli zachování souvislosti). Přidání hrany bude v podstatě analogické, až na to, že skončíme již v existujícím vrcholu. Odebrání hrany znamená přepočítání jejích sousedů, což je také triviální operace. Obdobně odebrání vrcholu znamená odebrání všech jeho hran a samotného vrcholu. Samotné odebraní bude fungovat pomocí mazacího modu. Pokud jej budete mít zapnutý, pak stačí kliknout na hranu či vrchol a pokud se neporuší souvislost, bude daný element odebrán.

Dále bude možné si nechat postupně vizualizovat další a další jednoduchá nakreslení K_n a zkoumat jejich vlastnosti. Odebrat z nich nějaký vrchol, hranu atd. jako v předchozím případě.

4.3 Vizualizace průsečíků a k-hran

V grafické části si bude moci uživatel zvolit, zdali chce vědět počty průsečíků, $\leq k$-hran atd. Průsečíky pro nás budou zvláštními vrcholy, proto vystoupit jejich počet bude snadné. Na určení počtu $E_k(D)$ bude třeba projít celý graf a spočítat počty orientovaných trojúhelníků. V závislosti na tom, jestli půjde o obecný graf budeme muset počítat buď oba směry, nebo jen jeden. Další hodnoty už jsou pouhými sumacemi $E_k(D)$, jak jsme si je zadefinovali. Také bude možnost vidět tyto hodnoty přes sousedy jednoho vrcholů. Poslední důležitá možnost bude při odebraní vrcholu či hrany zobrazí všechny invariantní hrany.

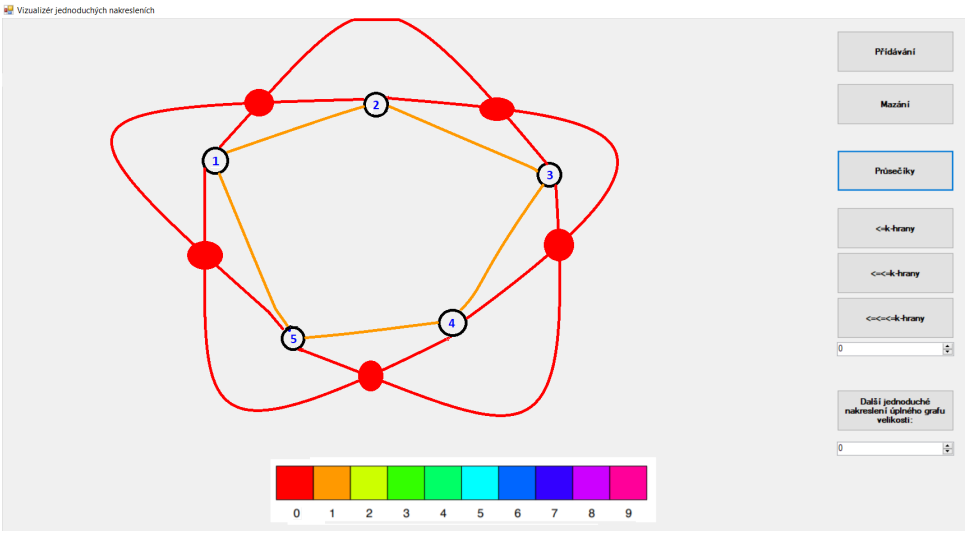
V aplikaci bude tedy 5 hlavních módů na vizualizaci vlastností, kde každá se bude volit tlačítkem. Vystoupení k-hran v barevném spektru bude vidět vždy. První bude zobrazení průsečíků. Ty se zobrazí jako červené body v grafu. Druha až čtvrtá možnost bude zobrazení všech $\leq k$-hran, $\leq\leq k$-hran, $\leq\leq\leq k$-hran, pomocí čárkovaní hran. Počet hran vrcholů grafu bude omezen 20, proto postačí barevné spektrum 10 barev. Barvy skutečných hodnot hran zůstanou nedotčené, jenom uvidíme pohromadě jejich společné vlastnosti. Pátá možnost bude zobrazení invariantních hran vůči odebraní vrcholu či hrany.

Hrana je *invariantní* k-hranou v nakreslení D vůči vrcholu v či hraně e, pokud zůstane k-hranou i v nakreslení, které vznikne z D odebraním vrcholu v či hrany e.

5 Obrazovky

5.1 Vizualizér

Na obrázku je příklad nakreslení grafu K_5 ve vizualizéru. Barevně jsou odlišené k-hrany dle palety dole. Po pravé straně jsou tlačítka jednotlivých módů vizualizéru. Dostupnými módy jsou **Přidávání**, **Mazání**, **Průsečíky**, **Počítání** $\leq$ k-**hran**, **Počítání** $\leq\leq$ k-**hran**, **Počítání** $\leq\leq\leq$ k-**hran** a mód **Invariantních hran**.



6 Time-line & Milestones

Datum	Milník	Způsob prezentace
30. dubna	Specifikace	Ukázka specifikace
17. května	Základní Half-Edge struktura pro práci s grafy	Ukázka kódu
15. červen	Testing základní **Half-Edge** struktury pro práci s grafy	Ukázka kódu
30. června	Celé generování	Ukázka kódu
31. července	Otestované generování	Ukázka kódu
31. srpna	Vizualizace	Ukázka programu
30. září	Hotový program	Ukázka programu