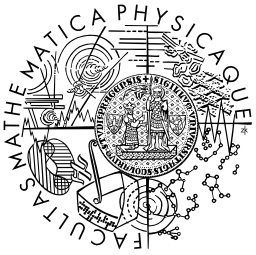


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Anna Gajdová

Jonesův polynom

Katedra algebry

Vedoucí bakalářské práce: doc. RNDr. Stanovský David, Ph.D.
Studijní program: Matematika
Studijní obor: obecná matematika

Praha 2018

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Chtěla bych tímto poděkovat doc. Davidu Stanovskému za odborné vedení této práce. Také bych chtěla poděkovat své rodině za podporu během mého studia.

Název práce: Jonesův polynom

Autor: Anna Gajdová

Katedra: Katedra algebry

Vedoucí bakalářské práce: doc. RNDr. Stanovský David, Ph.D., Katedra algebry

Abstrakt: Tématem této práce je Jonesův polynom daného uzlu a jeho výpočet. Nejprve definujeme Jonesův polynom dvěma způsoby: pomocí skein vztahů a pomocí závorkového polynomu a dokážeme ekvivalenci těchto definic. Dále na základě vztahu Jonesova a závorkového polynomu odvodíme algoritmus na jeho výpočet. Dokážeme, že algoritmus má časovou složitost $\mathcal{O}\left(2^{0,823n}\right)$, kde n značí počet křížení linkového diagramu. Nakonec shrneme výsledky testování algoritmu a jeho variant na datech. Algoritmus otestujeme mimo jiné na malých tabulkových uzlech, větších náhodných uzlech a torusových uzlech. U nejrychlejší varianty algoritmu odhadneme průměrnou časovou složitost výpočtu na náhodných uzlech $\mathcal{O}\left(2^{0,487n+o(n)}\right)$.

Klíčová slova: teorie uzlů, Jonesův polynom, uzlové invarianty

Title: Jones polynomial

Author: Anna Gajdová

Department: Department of Algebra

Supervisor: doc. RNDr. Stanovský David, Ph.D., Department of Algebra

Abstract: The subject of this thesis is the Jones polynomial of a given knot and its computation. First we define the Jones polynomial in two ways: using skein relations and using the bracket polynomial and we prove that these definitions are equivalent. Next we derive an algorithm for computation of the Jones polynomial based on its relation with the bracket polynomial. We prove that the time complexity of the algorithm is $\mathcal{O}\left(2^{0.823n}\right)$, where n denotes number of crossings in a link diagram. Lastly we present the results of running the algorithm and its variants on data. We test the algorithm among others on small table knots, bigger random knots and on torus knots. We estimate that the fastest variant of the algorithm runs on random knots with the average time complexity $\mathcal{O}\left(2^{0.487n+o(n)}\right)$.

Keywords: knot theory, Jones polynomial, knot invariants

Obsah

Úvod

Teorie uzlů se často zabývá otázkou, jak od sebe rozpoznat různé uzly či jak určit, jestli rozmotáním daného uzlu může vzniknout uzel triviální. Jedním ze způsobů, jak na tyto otázky odpovědět, je studium uzlových invariantů, tedy vlastností, které jsou stejné pro ekvivalentní uzly.
Užitečným druhem invariantů jsou invarianty polynomiální, mezi něž patří například Alexanderův nebo Conwayův polynom. V roce 1985 publikoval novozélandský matematik Vaughan Jones práci o novém polynomu, který objevil při studiu operátorových algeber [1]. Jeho výsledky měly velký vliv na vývoj oboru a objev nových druhů uzlových polynomů [2].
Tato práce se zabývá právě Jonesovým polynomem a algoritmem na jeho výpočet. Součástí práce je také implementace tohoto algoritmu a jeho testování.
Text práce je rozdělen do tří kapitol.
První kapitola je věnována definici Jonesova polynomu, jeho vlastnostem a ekvivalentní definici pomocí jiného uzlového polynomu, závorkového polynomu.
V druhé kapitole je na základě poznatků kapitoly první odvozen algoritmus na výpočet Jonesova polynomu a jeho varianty. Je zde také odhadnuta jeho výpočetní složitost.
V závěrečné kapitole jsou uvedeny výsledky testování algoritmu na datech. Algoritmus byl otestován mimo jiné na náhodných uzlech a lincích s větším počtem křížení a zvláštních typech uzlů. Jsou zde také srovnány rychlosti výpočtu různých variant algoritmu.

1. Definice a vlastnosti Jonesova polynomu

Cílem této kapitoly je definovat Jonesův polynom, dokázat jeho základní vlastnosti a popsat souvislost se závorkovým polynomem. Vycházíme z materiálů [2, 3, 4] a vypracování cvičení z těchto zdrojů.

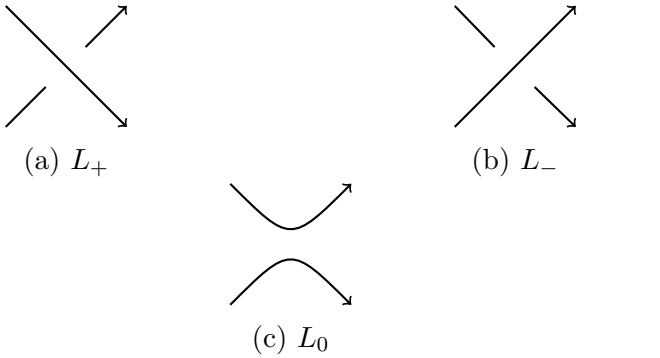
1.1 Základní pojmy

Matematický *uzel* je vnoření kružnice do trojrozměrného euklidovského prostoru. Dva uzly jsou *ekvivalentní*, pokud mezi nimi existuje homeomorfismus zachovávající orientaci. *Invariant* je zobrazení, které ekvivalentním uzlům přiřadí stejnou hodnotu. *Link* je více disjunktních či propletených uzlů. Pokud není řečeno jinak, pracujeme v textu s linky.
Při definování Jonesova polynomu je důležité rozlišovat mezi linkem a jeho *diagramem*. Diagram je vhodné rovinné nakreslení určité linkové projekce, v němž je rozlišeno, jestli křížení vedou *svrchu*, nebo *zdola*. *Triviální uzel* je uzel, k němuž existuje diagram bez křížení.
V diagramu *orientovaného* linku rozlišujeme křížení s *kladnou* a se *zápornou orientací*, viz obrázek 1.1.

(a) Kladná orientace. (b) Záporná orientace.

Obrázek 1.1: Orientace křížení.

Pro popis polynomů na uzlech a lincích se často používají *skein vztahy* [1]. Skein vztahy určují, jaká je spojitost mezi polynomy tří linků L_+, L_- a L_0, jejichž diagramy jsou identické až na oblast jednoho křížení. V linku L_+ má toto křížení kladnou orientaci, v L_- zápornou a v L_0 je křížení rozpojené, viz obrázek 1.2.



(a) L_+ (b) L_-

(c) L_0

Obrázek 1.2: Diagramy skein vztahu.

[1] česky přadenové vztahy

1.2 Definice Jonesova polynomu

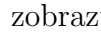
Definice 1. Jonesův polynom *orientovaného linku L je Laurentův polynom v proměnné* $\sqrt{t}$ *(tj. polynom v* $\mathbb{Z}\left[t^{1/2}, t^{-1/2}\right]$*), značený* $V_L(t)$ *, který*

1. *je linkový invariant,*
2. *je normalizovaný, tedy polynom* $V_{\circlearrowleft}$ *=1, kde* $\circlearrowleft$ *značí orientovaný triviální uzel,*
3. *splňuje skein vztah*

$$\frac{1}{t}V_{L_+} - tV_{L_-} = \left(\sqrt{t} - \frac{1}{\sqrt{t}}\right) V_{L_0}. \tag{1.1}$$

Lemma 1. *Buď L link, který se skládá z k neprotínajících se orientovaných triviálních uzlů. Pak pro Jonesův polynom linku L platí*

$$V_L(t) = \left(-\sqrt{t} - \frac{1}{\sqrt{t}}\right)^{k-1}.$$

Důkaz. Libovolně orientované triviální uzly jsou navzájem ekvivalentní. Vzorec tedy stačí dokázat pro diagram skládající se z k libovolně orientovaných disjunktních kružnic. Použijeme matematickou indukci.
Pro $k = 1$ vzorec platí podle druhé podmínky v definici 1.
Předvedeme i případ, kdy $k = 2$. Pak $L_0 =$  , $L_- =$  a $L_+ =$.
Diagramy L_+ a L_- zobrazují triviální uzly, takže $V_{L_+} = V_{L_-} = 1$. Použitím skein vztahu získáme

$$V_L = V_{L_0} = -\sqrt{t} - \frac{1}{\sqrt{t}}.$$

Pro $k > 2$ jsou L_- a L_+ diagramy linků s $k - 1$ kružnicemi. Z indukčního předpokladu a ze skein vztahu získáme vzorec

$$V_L = V_{L_0} = \left(-\sqrt{t} - \frac{1}{\sqrt{t}}\right)^{k-1}.$$

□

Poznámka. Z každého uzlového diagramu lze změnou několika křížení vedených shora na křížení vedených zdola získat diagram triviálního uzlu [3]. Z každého diagramu linku tedy můžeme změnou křížení získat diagram sjednocení triviálních uzlů, jejichž Jonesův polynom je podle předchozího lemmatu známý. Jonesův polynom každého linku lze tedy pomocí skein vztahu rekurzivně spočítat z jeho libovolného diagramu. Z toho plyno korektnost a jednoznačnost definice.

Definice Jonesova polynomu pomocí skein vztahů není vhodná pro algoritmický výpočet, neboť rozpoznat, jestli diagram odpovídá triviálnímu uzlu, je složitý problém. K výpočtu využijeme ekvivalentní definici založenou na použití *závorkového polynomu.*

1.3 Závorkový polynom

Závorkový polynom [2] je definován pouze pro diagramy neorientovaných linků, nikoli pro samotné linky.

Definice 2. Závorkový polynom *neorientovaného diagramu D, značený $\langle D\rangle$, je Laurentův polynom v proměnné A definovaný třemi odvozovacími pravidly:*

1. $\langle \bigcirc \rangle = 1$, *kde* $\bigcirc$ *značí diagram s jednou komponentou bez křížení,*
2. $\left\langle \times \right\rangle = A \left\langle)(\right\rangle + A^{-1} \left\langle \asymp \right\rangle$, *kde*
   - $\times$ *značí diagram obsahující toto křížení,*
   - $)($ *je diagram, který je s ním shodný až na dané křížení, které je zde* vertikální rozpojeno,
   - $\asymp$ *je diagram, v němž je křížení* rozpojeno horizontálně,
3. $\langle D \cup \bigcirc \rangle = (-A^2 - A^{-2}) \langle D \rangle$, *kde* $D \cup \bigcirc$ *značí sjednocení diagramu D a diagramu s jednou komponentou bez křížení.*

Poznámka. Pokud vztah v bodě *2.* předchozí definice otočíme o 90°, získáme vztah

$$\left\langle \times \right\rangle = A \left\langle \asymp \right\rangle + A^{-1} \left\langle)(\right\rangle .$$

Lemma 2. *Pro závorkové polynomy linků, jejichž diagramy obsahují smyčku, platí*

1. $\left\langle \text{⌒} \right\rangle = -A^{-3} \langle \;) \rangle$
2. $\left\langle \text{⌒} \right\rangle = -A^{3} \langle \;) \rangle$

Důkaz. Použitím odvozovacích pravidel dokážeme první bod.

$$\left\langle \text{⌒} \right\rangle = A \langle \;) \cup \bigcirc \rangle + A^{-1} \langle \;) \rangle = A(-A^2 - A^{-2}) \langle \;) \rangle + A^{-1} \langle \;) \rangle = -A^{-3} \langle \;) \rangle$$

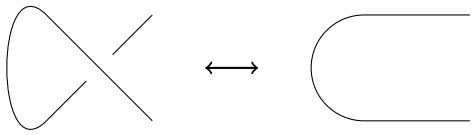
Druhý bod se dokáže analogicky.

□

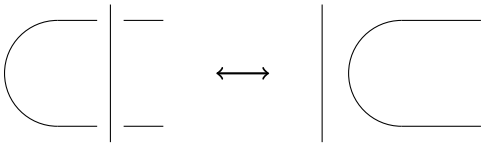
Dva diagramy znázorňují ekvivalentní linky, pokud lze jeden získat z druhého sérií Reidemeisterových pohybů [3], viz obrázek 1.3. Z lemmatu 2 plyne, že závorkový polynom není invariantní vůči Reidemeisterovu pohybu typu I. Ukážeme, že je invariantní vůči Reidemeisterovým pohybům typu II a typu III.

Tvrzení 3. *Závorkový polynom je invariantní vůči Reidemeisterovým pohybům typu II a III.*

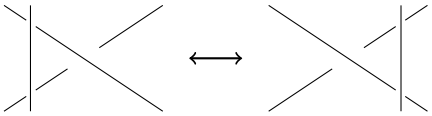
[2] anglicky bracket polynomial nebo Kauffman bracket



(a) Typ I.



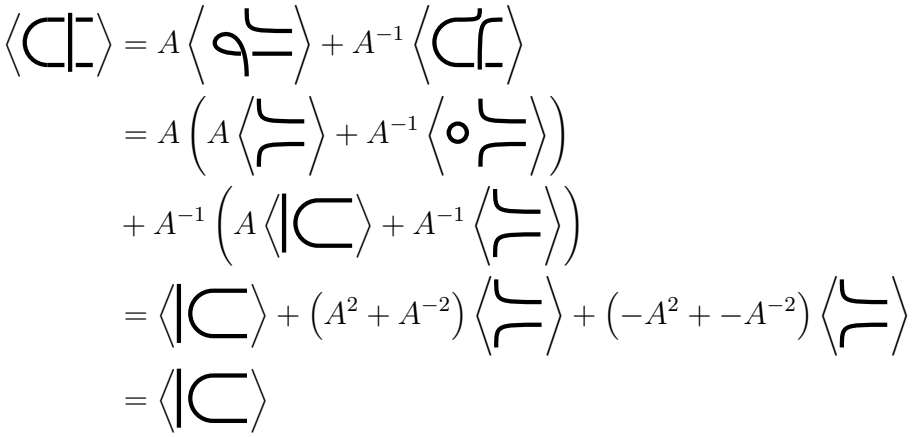
(b) Typ II.



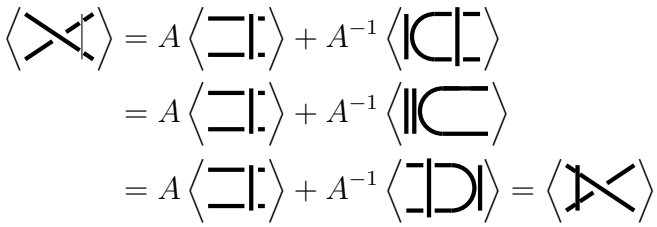
(c) Typ III.

Obrázek 1.3: Reidemeisterovy pohyby.

Důkaz. Použitím odvozovacích pravidel dokážeme invarianci vůči pohybu typu II.



Invariance vůči pohybu typu III plyne z invariance vůči pohybu typu II.



□

Aby byl závorkový polynom invariantní také vůči Reidemeisterovu pohybu typu I, je nutné vynásobit jej výrazem, který vyjadřuje míru zakroucení.

Definice 3. *Buď D orientovaný diagram.* Zakroucení *(writhe)* $w(D) \in \mathbb{Z}$ *je součet znamení orientací všech křížení v D.*

Lemma 4. *Zakroucení je invariantní vůči Reidemeisterovým pohybům typu II a typu III.*

Důkaz. Uvažujme ta dvě křížení, která jsou v diagramu odstraněna nebo vzniknou pohybem typu II či pohybem typu III. Jedno křížení musí vždy být kladné a druhé záporné orientace. Jejich odstranění tedy neovlivní hodnotu zamotání. □

Definice 4. Normalizovaný závorkový polynom $X_L(A)$ *orientovaného linku* L *definujeme*

$$X_L(A) = \left(-A^{-3}\right)^{-w(L_+)} \langle D \rangle,$$

kde D *značí libovolný diagram linku* L*.*

Poznámka. Definice je korektní, neboť následující tvrzení ukazuje, že nezáleží na volbě diagramu.

Tvrzení 5. *Normalizovaný závorkový polynom je linkový invariant.*

Důkaz. Závorkový polynom i zakroucení jsou podle tvrzení 3 a lemmatu 4 invariantní vůči Reidemeisterovým pohybům typu II a III, invariantní je tedy i jejich součin.

Invariance vůči typu I plyne z lemmatu 2 a faktu, že křížení  je vždy kladné a křížení vždy záporné.

□

Tvrzení 6. *Při substituci proměnné* $A = t^{-1/4}$ *je normalizovaný závorkový polynom* $X_L(A)$ *roven Jonesovu polynomu* $V_L(t)$*.*

Důkaz. Ověříme, že $X_L\left(t^{-1/4}\right)$ splňuje podmínky v definici 1:

1. Normalizovaný závorkový polynom je podle předchozího tvrzení linkový invariant.

2. Pro zamotání triviálního uzlu platí $w\left(\bigcirc\right) = 0$, tedy

$$X_{\bigcirc} = \left(-A^3\right)^{w(\bigcirc)} \langle \bigcirc \rangle = 1.$$

3. Dokážeme ekvivalentní tvrzení, že $X_L(A)$ splňuje skein vztah 1.1 při substituci $t = A^{-4}$.
   Buď L_+, L_- a L_0 diagramy, pak platí

$$\left(-A^{-3}\right)^{-w(L_\pm)} = -A^{\mp 3}\left(-A^{-3}\right)^{-w(L_0)}$$

   a dostáváme

$$\begin{aligned} X_{L_+}(A) &= \left(-A^{-3}\right)^{-w(L_+)} \langle L_+ \rangle = -A^{-3}\left(-A^{-3}\right)^{-w(L_0)} \left\langle \times \right\rangle \\ &= -A^{-3}\left(-A^{-3}\right)^{-w(L_0)} \left(A \left\langle \times \right\rangle + A^{-1} \left\langle) (\right\rangle\right) \end{aligned}$$

$$X_{L_-}(A) = \left(-A^{-3}\right)^{-w(L_-)} \langle L_- \rangle = -A^3 \left(-A^{-3}\right)^{-w(L_0)} \left\langle \times \right\rangle$$
$$= -A^3 \left(-A^{-3}\right)^{-w(L_0)} \left(A \left\langle)(\right\rangle + A^{-1} \left\langle \asymp \right\rangle\right).$$

Levá strana vztahu 1.1 se tedy při substituci $t = A^{-4}$ rovná

$$A^4 X_{L_+}\left(t^{1/4}\right) - A^{-4} X_{L_-}\left(t^{1/4}\right) = \left(-A^3\right)^{-w(L_0)} \left[-A \left(A \left\langle \asymp \right\rangle + A^{-1} \left\langle)(\right\rangle\right)\right.$$
$$\left. + A^{-1} \left(A \left\langle)(\right\rangle + A^{-1} \left\langle \asymp \right\rangle\right)\right]$$
$$= \left(-A^3\right)^{-w(L_0)} \left(A^{-2} - A^2\right) \left\langle \asymp \right\rangle$$
$$= \left(A^{-2} - A^2\right) X_{L_0},$$

což jsme měli dokázat.

□

2. Výpočet Jonesova polynomu

V této kapitole se budeme zabývat problémem výpočtu Jonesova polynomu. Odvodíme algoritmus na výpočet a odhadneme jeho časovou složitost.

2.1 Výpočetní složitost problému

Je dokázáno, že problém výpočtu Jonesova polynomu alternujího uzlu je #P-těžký [5].

Třída #P obsahuje problémy určení počtu přijímacích cest nedeterministického Turingova stroje. Jedná se o rozšíření problémů třídy NP, místo otázky, jestli problém má řešení, se ptáme, kolik řešení vůbec existuje. Zástupcem této třídy je #SAT, tedy problém určit, kolik existuje pravdivostních ohodnocení Boolovské formule. Dalším problémem této třídy je, jak spočítat, kolik existuje perfektních párování v bipartitním grafu [6].

Není znám polynomiální algoritmus, který by řešil NP-těžký problém, tedy ani #P-těžký problém.

2.2 Algoritmus

Náš algoritmus na výpočet Jonesova polynomu předpokládá na vstupu diagram orientovaného linku zapsaný v PD notaci. Počet křížení diagramu označíme n.

2.2.1 PD notace

PD[1] notace je zápis sestávající z n čtveřic čísel, pro každé křížení jedno. Notace jednoznačně popisuje daný diagram. Zápis diagramu v PD notaci se získá následovně: úseky mezi kříženími se očíslují po směru orientace linku čísly od 1 do $2n$. Každé křížení se označí čtyřmi přilehlými úseky, přičemž se začne úsekem, který do křížení vstupuje zdola, a pokračuje se s úseky navazujícími proti směru hodinových ručiček, viz obrázek 2.1.

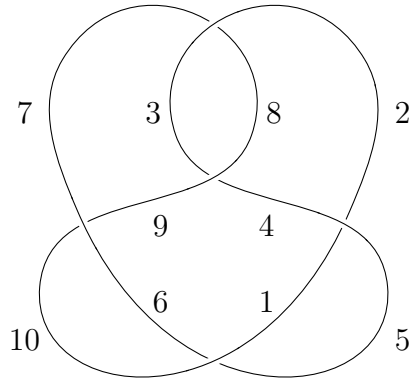
Pro nejrychlejší práci s diagramem v PD notaci si pro každý úsek pamatujeme čtveřice popisující ta křížení, která mu náleží.

2.2.2 Výpočet Jonesova polynomu ze závorkového polynomu

K výpočtu Jonesova polynomu použijeme závorkový polynom. Podle tvrzení 6 se Jonesův polynom získá z normalizovaného závorkového polynomu substitucí proměnné. Pokud tedy známe závorkový polynom daného diagramu, stačí vypočítat zamotání a provést substituci proměnné, viz algoritmus 1.

V PD notaci lze rychle určit, jestli je křížení kladné, či záporné orientace. Zamotání tedy spočítáme v lineárním čase vzhledem k počtu křížení. Z toho plyne,

[1] planar diagram



Obrázek 2.1: Uzel s PD notací [1, 5, 2, 4], [7, 3, 8, 2], [3, 9, 4, 8], [5, 1, 6, 10], [9, 7, 10, 6].

Algoritmus 1: Výpočet Jonesova polynomu ze závorkového polynomu.

Data: Diagram D linku L s n kříženími
Result: Jonesův polynom linku L v proměnné t

závorkovýPolynom(A) ← `Bracket(D)` `/* v proměnné A */`
zamotání ← `Writhe(D)`
normalizovanýPolynom(A) ← $(-A^3)^{-\text{zamotání}} \times$ závorkovýPolynom(A)
jonesůvPolynom(t) ← `Substituce(`normalizovanýPolynom(A)`,` $A \to t^{1/4}$`)`

return jonesůvPolynom(t)

že výpočet Jonesova polynomu ze závorkového polynomu má lineární časovou složitost.

Dále se budeme zabývat algoritmem na výpočet závorkového polynomu.

2.2.3 Přímočarý výpočet závorkového polynomu

Definice 5. Rozpojením křížení *diagramu D rozumíme rozpojení v horizontálním a vertikálním směru ve smyslu bodu 2 v definici 2 závorkového polynomu. Rozpojením diagramu s n kříženími vzniknou dva* synové*, oba diagramy s $n-1$ kříženími.*

Z definice 2 závorkového polynomu plyne jednoduchý rekurzivní algoritmus na jeho výpočet. Pokud diagram D nemá žádná křížení, má polynom hodnotu 1. Pokud má diagram $n \geq 1$ křížení, jedno se náhodně zvolí. Rozpojením tohoto křížení horizontálně a vertikálně vzniknou dva synové D_h a D_v. Závorkový polynom diagramu D se spočítá ze závorkových polynomů jeho synů.
V PD notaci nejsme schopni zaznamenat, jestli při rozpojení křížení vznikla disjunktní kružnice, diagramy synů jsou tedy bez nich. Pokud při rozpojování disjunktní kružnice vznikla, musíme spočtený závorkový polynom syna vynásobit členem $-A^2 - A^{-2}$, podrobnosti viz algoritmus 2.
Algoritmus se vždy zastaví a má časovou složitost $\mathcal{O}(2^n)$. Stejnou časovou složitost má i výpočet Jonesova polynomu používající tento způsob výpočtu závorkového polynomu.

2.2.4 Průběžné rozmotávání

Algoritmus na výpočet závorkového polynomu zrychlíme, pokud se link pokusíme průběžně rozmotávat, tedy pokud nalezneme diagram ekvivalentního linku s menším množstvím křížení.

Definice 6. *Diagram je* rozmotaný*, pokud v něm nelze odstranit křížení použitím Reidemeisterova pohybu typu I či II. V opačném případě řekneme, že diagram je možné* rozmotat*.*

V PD notaci lze v lineárním čase vzhledem k počtu křížení rozpoznat případy, kdy je možné link rozmotat.

Reidemeisterovým pohybem typu I se zbavíme jednoho křížení, ovšem výsledný závorkový polynom se podle lemmatu 2 změní o násobek $-A^{\pm 3}$.

Reidemeisterovým pohybem typu II se zbavíme dvou křížení a závorkový polynom zůstane podle tvrzení 3 stejný. Více viz algoritmus 3.

Rozmotávání běží v lineárním čase vzhledem k n, celková časová složitost tedy zůstává $\mathcal{O}(2^n)$.

2.2.5 Vhodná volba křížení

Křížení k rozpojení jsme volili náhodně, algoritmus se pokusíme zlepšit vhodnou volbou křížení, aby bylo diagram možné průběžně rozmotávat. V této sekci dokážeme následující lemma.

Lemma 7. *Ke vzniku diagramu, který lze rozmotat, stačí vždy rozpojit nejvýše dvě křížení.*

Algoritmus bude volit právě ta křížení, jejichž rozpojení u synů (nebo vnuků) zajistí možnost rozmotání.

Diagram jako rovinný graf

Každý linkový diagram odpovídá rovinnému grafu, v němž křížení představují vrcholy (vždy stupně čtyři) a úseky mezi kříženími hrany. Diagram s n kříženími odpovídá grafu s n vrcholy a $2n$ hranami. Dále budeme v této sekci k popisu diagramů používat grafovou terminologii.

Lemma 8. *V rozmotaném diagramu existuje stěna ohraničená méně než čtyřmi hranami.*

Důkaz. Podle Eulerova vzorce pro rovinné grafy platí

$$v - e + f = 2,$$

kde v značí počet vrcholů, e počet hran a f počet stěn (důkaz např. v [7]).
V našem případě tedy dostáváme vzorec pro počet stěn v linkovém diagramu

$$f = n + 2$$

.
Každá hrana diagramu náleží dvěma stěnám, rozdělujeme tedy $4n$ hran mezi $n + 2$ stěn. Z toho plyne, že musí existovat stěna, která je ohraničená méně než čtyřmi hranami.

□

Algoritmus 2: Výpočet závorkového polynomu.

Function `Bracket(D)`

Data: Diagram D linku L s n kříženími
Result: Závorkový polynom diagramu D v proměnné A

if *diagram D nemá žádná křížení* **then**
  return 1

zvol náhodně křížení diagramu D

Dh ← link D, kde křížení je rozpojeno horizontálně
Dv ← link D, kde křížení je rozpojeno vertikálně

if *v linku* Dh *vznikla disjunktní kružnice* **then**
  $k_h \leftarrow 1$
else
  $k_h \leftarrow 0$
if *v linku* Dv *vznikla disjunktní kružnice* **then**
  $k_v \leftarrow 1$
else
  $k_v \leftarrow 0$

$$\text{závorkovýPoly}(t) \leftarrow A(-A^2 - A^{-2})^{k_h} \texttt{ Bracket(Dh)} + A^{-1}(-A^2 - A^{-2})^{k_v} \texttt{ Bracket(Dv)}$$

return závorkovýPoly(t)

Algoritmus 3: Výpočet závorkového polynomu s rozmotáváním

Function `Bracket(D)`

rozmotej diagram D Reidemeisterovým pohybem typu I

if *něco rozmotáno* **then**
  e ← součet znamení rozmotaných křížení
  k ← počet vzniklých disjunktních kružnic
  return $(-A^2 - A^{-2})^{\mathsf{k}}(-A^{-3\mathsf{e}})$ `Bracket(`*rozmotaný diagram*`)`

rozmotej diagram D Reidemeisterovým pohybem typu II

znovu rozmotej diagram D Reidemeisterovým pohybem typu I

if *něco rozmotáno typem I* **then**
  e ← součet znamení rozmotaných křížení
  k ← počet vzniklých disjunktních kružnic
  return $(-A^2 - A^{-2})^{\mathsf{k}}(-A^{-3\mathsf{e}})$ `Bracket(`*rozmotaný diagram*`)`
.
. `/* Dále jako algoritmus 2. */`
.

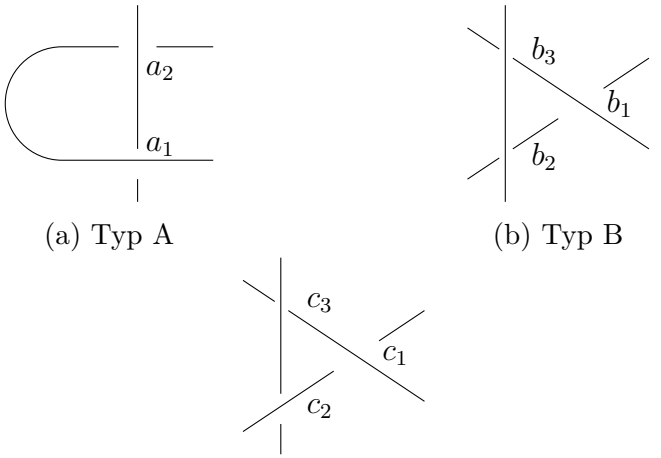
return závorkovýPoly

Typy stěn

Popíšeme, jak mohou v rozmotaném diagramu vypadat stěny ohraničené méně než čtyřmi hranami.

Stěna s jednou hranou by v diagramu odpovídala smyčce, ty jsou už odstraněny rozmotáním.

Možné stěny s dvěma a třemi hranami jsou zobrazené na obrázku 2.2.



(a) Typ A (b) Typ B

(c) Typ C

Obrázek 2.2: Typy stěn ohraničených třemi a méně hranami.

Stěna se dvěma hranami, kterou není možné rozmotat druhým Reidemeisterovým pohybem, musí v diagramu odpovídat typu A. Všimněme si, že rozpojením křížení a_1 i a_2 buď ve vertikálním, nebo horizontálním směru vznikne smyčka. Volbou křížení a_1 nebo a_2 je tedy zaručeno, že jeden syn má po rozmotání nejvýše $n-2$ křížení.

Stěna se třemi hranami v diagramu odpovídá buď typu B, nebo C.

Ve stěně typu B získáme vertikálním rozpojením křížení b_1 syna, jenž jde rozmotat druhým Reidemeisterovým pohybem. Jeho diagram tedy bude mít po rozmotání nejvýše $n-3$ křížení.

Ve stěně typu C není rozpojením žádného křížení rozmotání zaručeno. Ovšem rozpojením libovolného křížení získáme jednoho syna se stěnou typu A. Existuje tedy vnuk s nejvýše $n-3$ kříženími.

Tímto jsme dokázali lemma 7.

2.2.6 Výsledný algoritmus pro výpočet závorkového polynomu

Snažíme se zajistit možnost rozmotání u synů, budeme tedy při rozpojování preferovat křížení ve stěnách A a B před stěnou C. Dále rozlišíme dvě varianty výsledného algoritmu: *algoritmus A* navíc preferuje křížení a_i před b_1, *algoritmus B* preferuje b_1 před a_i. Celý postup je shrnut v algoritmu 4.

Původní algoritmus 3 s náhodnou volbou křížení a rozmotáváním budeme označovat *algoritmus RND*.

V PD notaci je možné vhodné křížení stěn A, B i C nalézt v lineárním čase vzhledem k n, časová složitost tedy zůstává $\mathcal{O}(2^n)$. V následující sekci 2.3 se

pokusíme získat lepší odhady složitosti.

Obě varianty výsledného algoritmu jsou podle úvah v předchozích částech korektní a vždy se zastaví.

2.2.7 Implementace algoritmu

Algoritmus jsme implementovali v programovacím jazyce Python.

Práce s diagramem v PD notaci s využitím vhodných datových struktur je sice rychlá, ale náročná na rozbor všech možných případů. Ačkoli je tedy samotný algoritmus poměrně přímočarý, délka kódu narostla kvůli nutnosti rozlišení všech případů při rozpojování křížení, rozmotávání diagramu a hledání vhodného křížení.

2.3 Analýza složitosti algoritmu

V této sekci získáme horní a dolní odhad časové složitosti algoritmu na výpočet závorkového a Jonesova polynomu. V následující analýze nezáleží, jestli se jedná o variantu A, nebo B.

Rychlost algoritmu je závislá na počtu křížení n a na míře rozmotávání. Například na diagramu, který je možné rozmotat absolutně, poběží nejhůř v kvadratickém čase vzhledem k n. V kvadratickém čase také algoritmus běží na jistém typu torusových uzlů, viz sekce 3.3.

2.3.1 Horní odhad

Tvrzení 9. *Výpočet Jonesova polynomu založený na algoritmu 4 na výpočet závorkového polynomu má v nejhorším případě časovou složitost* $\mathcal{O}\left(2^{0,823n}\right)$.

Důkaz. Horní odhad rychlosti algoritmu provedeme na příkladu diagramu, v němž dojde pouze k minimálnímu rozmotávání. Nedojde tedy k rozmotání jiných křížení než těch, u kterých je to zaručeno volbou křížení k rozpojení, viz rozbor v sekci 2.2.5. Také v případech, kdy není předchozí volbou křížení zaručena existence stěny typu A, bude existovat pouze stěna typu C, která zaručuje rozmotání pouze u vnuka. V tomto případě tedy nezáleží, jestli se jedná o variantu algoritmu A, nebo B.

Průběh algoritmu zakreslíme formou binárního stromu, kde číslo ve vrcholu značí počet křížení diagramu. V kořeni máme diagram s n kříženími, v němž rozmotáme křížení stěny C, takže jeho synové jsou diagramy s $n-1$ kříženími a obsahují stěnu typu A. Jejich synové budou diagramy velikosti $n-2$ a $n-3$. V lichých hladinách stromu má tedy vrchol syny o jedno křížení menší, v sudých hladinách je jeden syn o dvě křížení menší.

Algoritmus 4: Výsledný algoritmus pro výpočet závorkového polynomu, varianty A a B.

Function `Bracket(D)`

Data: Diagram D linku L s n kříženími
Result: Závorkový polynom v proměnné A

if *diagram* D *nemá žádná křížení* **then**
  return 1

rozmotej diagram jako v algoritmu 3

if *varianta A* **then**
  if *existuje stěna typu A* **then**
    křížení $\leftarrow a_1$
  else
    if *existuje stěna typu B* **then**
      křížení $\leftarrow b_1$
    else
      křížení $\leftarrow c_1$

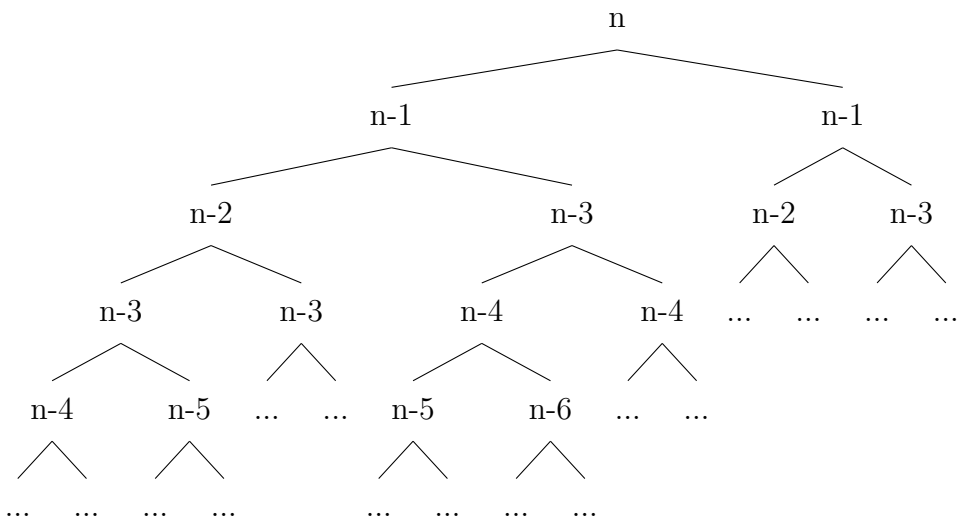
if *varianta B* **then**
  if *existuje stěna typu B* **then**
    křížení $\leftarrow b_1$
  else
    if *existuje stěna typu A* **then**
      křížení $\leftarrow a_1$
    else
      křížení $\leftarrow c_1$

Dh $\leftarrow$ link D, kde křížení je rozpojeno horizontálně
Dv $\leftarrow$ link D, kde křížení je rozpojeno vertikálně

if *v linku* Dh *vznikla disjunktní kružnice* **then**
  $k_h \leftarrow 1$
else
  $k_h \leftarrow 0$
if *v linku* Dv *vznikla disjunktní kružnice* **then**
  $k_v \leftarrow 1$
else
  $k_v \leftarrow 0$

$$\text{závorkovýPoly}(t) \leftarrow A(-A^2 - A^{-2})^{k_h}\ \texttt{Bracket(Dh)} + A^{-1}(-A^2 - A^{-2})^{k_v}\ \texttt{Bracket(Dv)}$$

return závorkovýPoly(t)



Získali jsme následující rekurentní rovnici na počet operací výpočtu závorkového polynomu diagramu velikosti n.

$$T(n) = 2T(n-2) + 2T(n-3) + c_l n + 2c_l(n-1) \tag{2.1}$$

Člen $c_l n + 2c_l(n-1)$, kde c_l je konstanta, vyjadřuje, že na rozdělení každého diagramu na syny (rozmotání a volba křížení) je potřeba lineární počet operací vzhledem k počtu křížení.

Rovnici vyřešíme podle postupu uvedeného v [7]. Nejprve nalezneme vytvořující funkci $F(x)$ posloupnosti $(T_n)_0^\infty$, tedy funkci, pro kterou platí

$$F(x) = \sum_{n=0}^{\infty} T(n)x^n.$$

V našem případě dostaneme z rekurentního vztahu 2.1 rovnici

$$F(x) - x^2 - x - 1 = 2x^2F(x) + 2x^3F(x) - \frac{x^3(4x-7)}{(x-1)^2}$$

a jejím vyřešením

$$F(x) = \frac{-1 + x + 2x^2 - 10x^3 + 5x^4}{(1-x)^2(-1 + 2x^2 + 2x^3)}.$$

Označme $r_1, r_2, \bar{r_2}$ kořeny polynomu $-1 + 2x^2 + 2x^3$. Rozklad funkce $F(x)$ na parciální zlomky je

$$F(x) = \frac{a_1 x}{1 - \frac{1}{r_1}x} + \frac{a_2 x + b_2}{\left(1 - \frac{1}{|r_2|}x\right)^2} + \frac{a_3 x + b_3}{(1-x)^2}$$

pro jisté konstanty a_i, b_i.
Platí

$$\frac{1}{r_1} = \frac{6}{-2 + \left(46 - 6\sqrt{57}\right)^{1/3} + \left(46 - 6\sqrt{57}\right)^{1/3}} \approx 2^{0,823}$$

$$\frac{1}{|r_2|} \approx 2^{0,088}.$$

Vzorec pro T_n získáme rozvinutím $F(x)$ do mocninné řady. To provedeme využitím známých vztahů

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$$
$$\frac{1}{(1-x)^2} = \sum_{n=0}^{\infty} (n+1)x^n$$

a získáme výsledný vzorec

$$T(n) \approx c_1 2^{0{,}823n} + c_2 n 2^{0{,}088n} + c_3 2^{0{,}088n} + c_4 n + c_5 \tag{2.2}$$

kde c_i jsou jisté konstanty.

Ze vzorce 2.2 plyne

$$T(n) = \mathcal{O}\left(2^{0{,}823n}\right).$$

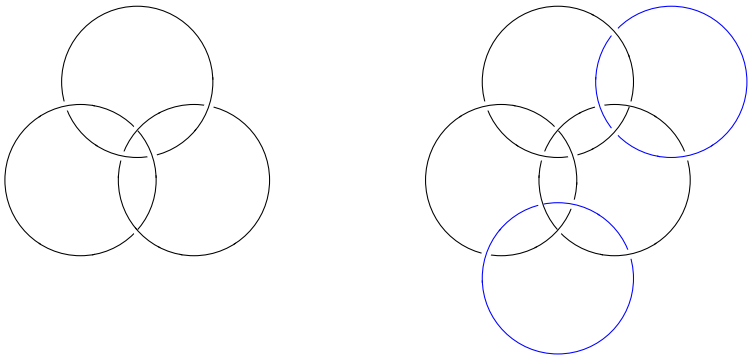
Podle sekce 2.2.2 je na výpočet Jonesova polynomu ze závorkového polynomu potřeba pouze lineární počet operací. Dohromady má tedy výpočet Jonesova polynomu časovou složitost $\mathcal{O}\left(2^{0{,}823n}\right)$.

□

2.3.2 Dolní odhad

Tvrzení 10. *Výpočet Jonesova polynomu založený na algoritmu 4 na výpočet závorkového polynomu má v nejhorším případě časovou složitost* $\Omega\left(2^{0{,}75n}\right)$.

Důkaz. Provedeme analýzu průběhu algoritmu 4 na linku s diagramem, který vznikne z diagramu Borromeovských kruhů přidáváním kružnic. Kružnici přidáváme tak, aby vždy protla dva další kruhy celkem ve čtyřech bodech a upravíme křížení, aby vzniklý diagram byl alternující, tj. aby se střídala křížení vedená zdola a shora, viz obrázek 2.3. Diagram složený z k kružnic označíme B_k.

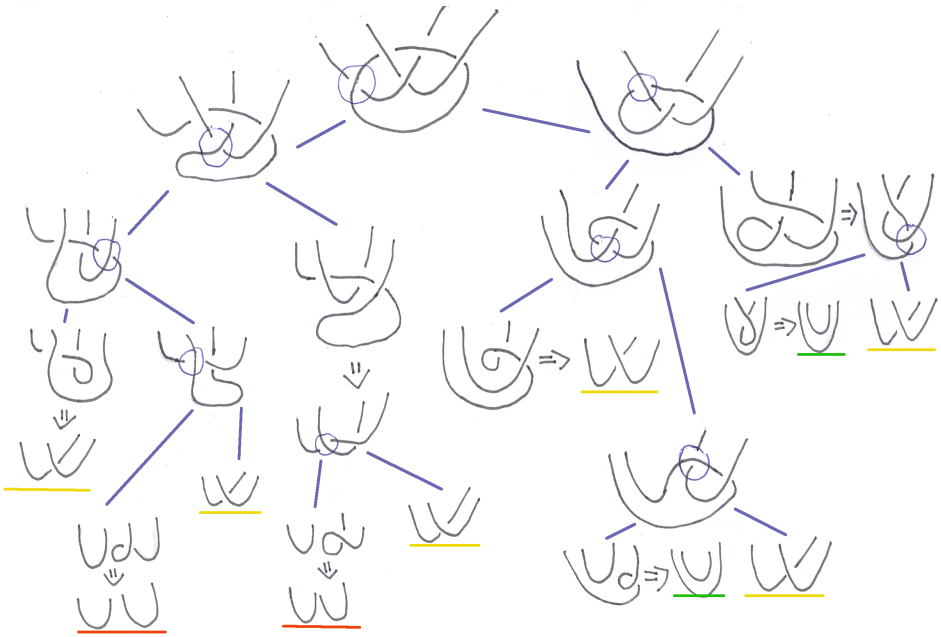


(a) Borromeovské kruhy (b) Přidání dvou kružnic

Obrázek 2.3: Vznik diagramu B_k z důkazu tvrzení 10.

Diagram B_k má $4k-6$ křížení, můžeme tedy tímto způsobem získat diagram s počtem křížení větším než libovolné n. Každá stěna diagramu B_k je typu C.

Na obrázku 2.4 je znázorněn strom s průběhem algoritmu na diagramu B_k takový, že jsou postupně odstraňovány krajní kružnice. V kořeni se nachází úsek diagramu obsahující nějakou krajní kružnici. Zakroužkováním je označeno křížení, které bylo vybráno k rozpojení (volba křížení odpovídá variantám A i B). Šipkou je znázorněno, pokud dojde k rozmotání diagramu. Žlutě podtržení potomci jsou diagramy B_{k-1}. Červeně a zeleně podržení potomci jsou synové diagramu B_{k-1}, jak je vidět na obrázku 2.5.



Obrázek 2.4: Průběh algoritmu na diagramu B_k.

Obrázek 2.5: Synové diagramu B_{k-1}.

Dohromady tedy z diagramu B_k rozpojováním vznikne šest diagramů B_{k-1} a čtyři synové B_{k-1}. Výpočet závorkového polynomu diagramu B_k je tedy těžký alespoň jako výpočet polynomu osmi diagramů B_{k-1}.

Počet operací výpočtu závorkového polynomu diagramu s n kříženími tedy splňuje pro jisté kladné konstanty c_0, c_1 rekurenci

$$T(n) = 8T(n-4) + c_1 n + c_0 \geq 8T(n-4).$$

Z toho plyne, že $T(n) = \Omega\left(8^{\frac{n}{4}}\right) = \Omega\left(2^{0,75n}\right)$.

Na výpočet Jonesova polynomu ze závorkového polynomu je potřeba lineární počet operací, tedy i výpočet Jonesova polynomu má časovou složitost $\Omega\left(2^{0,75n}\right)$. □

3. Testování algoritmu na datech

V této části shrneme výsledky implementace výpočtu Jonesova polynomu na malých uzlech, náhodých uzlech a na speciálních typech uzlů. Budeme se zabývat algoritmy A, B a RND, které byly popsány v sekci 2.2.6.

3.1 Malé tabulkové uzly

Algoritmy jsme vyzkoušeli na všech uzlech, které mají projekci s maximálně dvanácti křiženími. PD notace uzlů byla získána z databáze KnotInfo [8]. U všech tří algoritmů byl už u malých uzlů znatelný exponenciální růst doby běhu, viz graf 3.1.

Ze srovnání průměrných dob běhu plyne, že nejlepšího času dosahuje algoritmus A, těsně následuje algoritmus RND a nejpomalejší je algoritmus B, viz graf 3.2.

3.2 Náhodné uzly a linky

3.2.1 Generování náhodných linků a uzlů

Náhodné linky jsme generovali pomocí rovinných grafů, neboť mezi linky a rovinnými grafy existuje vzájemně jednoznačná korespondence [3]. (Jedná se o jinou korespondenci než mezi linky a rovinnými grafy s vrcholy stupně čtyři popsanou v sekci 2.2.5).

Převod rovinného grafu na link

Rovinný graf s n hranami odpovídá linku s n křiženími. Každé hraně přiřadíme náhodně kladné, či záporné znamení a umístíme na ni křížení příslušného typu. Úseky mezi křiženími jsou tím již jednoznačně určené: musí spojovat křížení mezi nejbližšími hranami tak, aby nevznikla žádná další křížení, viz obrázek 3.3b.

Generování náhodných rovinných grafů

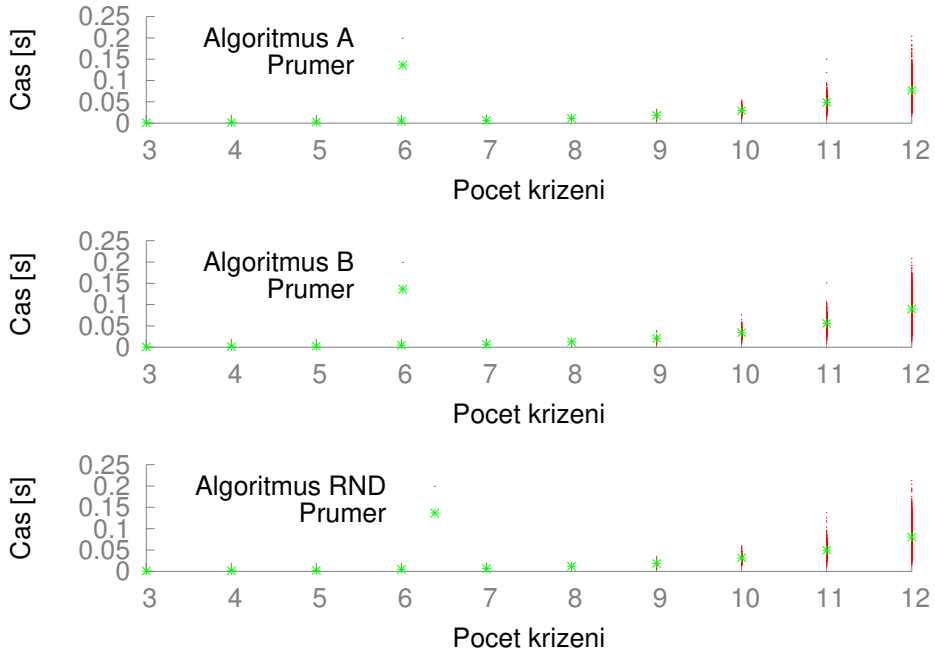
Graf s n hranami získáme následujícím způsobem. Vygenerujeme vhodný počet náhodných bodů v rovině a nalezeneme jejich triangulaci, tj. spojíme body hranami tak, aby byl polygon tvořící konvexní obal bodů rozdělen na trojúhelníky, viz obrázek 3.3a.

Získali jsme tak rovinný graf s původními body jako vrcholy. Pokud je počet hran menší než n, provedeme triangulaci znovu s větším počtem bodů. Pokud je počet hran větší než n, odstraníme potřebný počet náhodně zvolených hran.

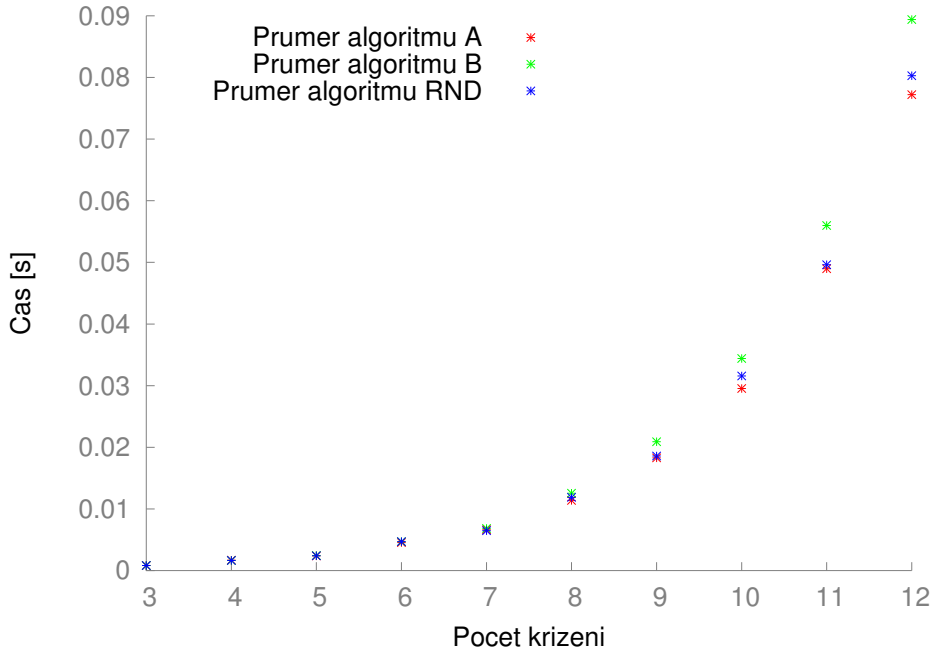
Implementace

Triangulace bodů je snadno implementovatelný geometrický problém.

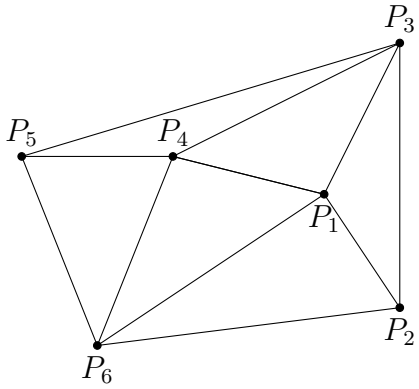
Se získaným rovinným grafem pracujeme jako s množinou vrcholů a k nim příslušným hranám. Hrany jsou seřazeny podle pořadí, jak k danému vrcholu



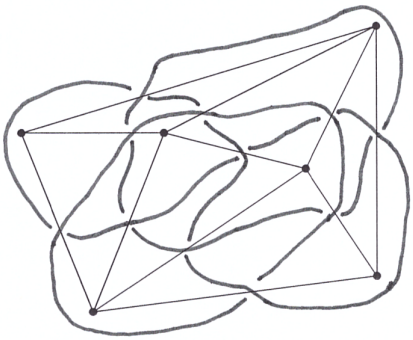
Obrázek 3.1: Graf dob běhu algoritmů na tabulkových uzlech do 12 křížení s vyznačenými průměry.



Obrázek 3.2: Graf průměrných dob běhu algoritmů na tabulkových uzlech do 12 křížení.



(a) Triangulace šesti bodů.



(b) Link vzniklý z triangulace.

Obrázek 3.3: Převod rovinného grafu na link.

v nakreslení grafu přiléhají. Z této struktury je již možné získat PD notaci příslušného linku.

V PD notaci lze procházkou po vláknu snadno poznat, jestli je vygenerovaný link uzlem. Také jsme jednoduchou operací schopni uzel změnit na alternující, tedy takový, v němž se střídají křížení vedená zdola a svrchu.

Dokážeme tedy nagenerovat uzly, alternující uzly a linky libovolné velikosti.

Takto generované uzly jsou většinou buď rozmotané, nebo lze rozmotáním odstranit pouze několik křížení.

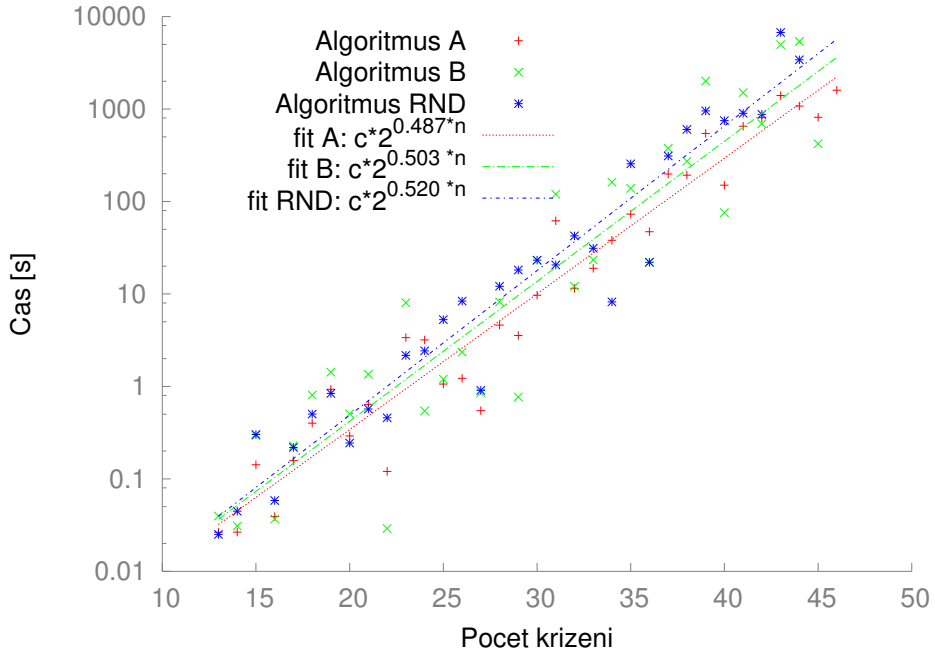
3.2.2 Test

Rychlost algoritmů jsme otestovali na uzlech (graf 3.4), alternujících uzlech (graf 3.5) a lincích (graf 3.6) s počtem křížení $n \leq 46$.

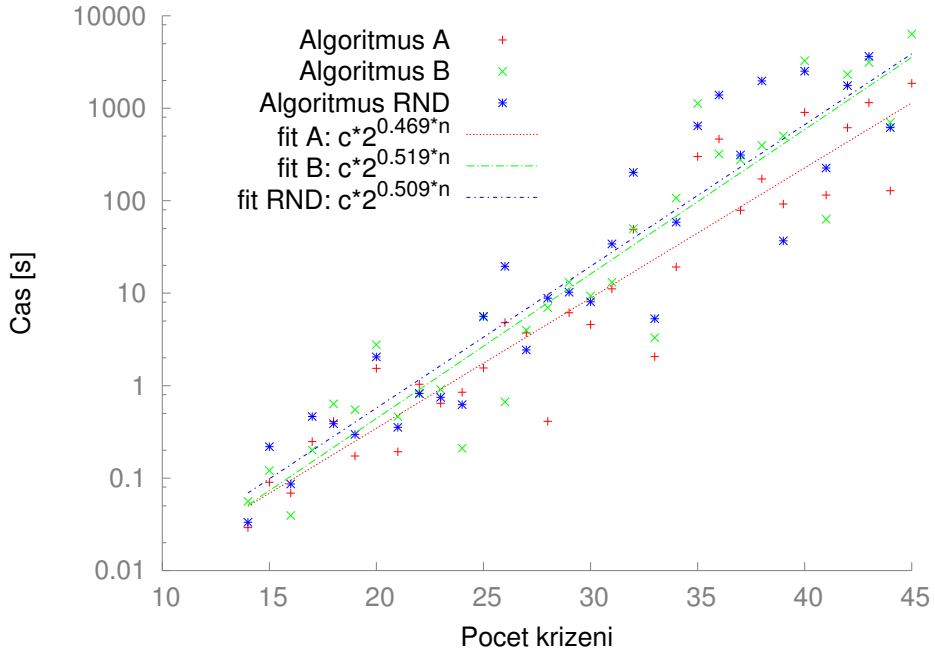
Pokusíme se u jednotlivých algoritmů v závislosti na druhu testovaných dat odhadnout průměrnou časovou složitost $T(n)$. Na základě odhadů složitosti dokázaných v sekci 2.3 budeme předpokládat, že $T(n) = \mathcal{O}\left(2^{kn+o(n)}\right)$ pro jistý koeficient k.

Označme $t(n) = \log_2(T(n)) = \mathcal{O}\left(kn + o(n)\right)$. Sublineární členy nahradíme konstantou a odhad koeficientu k získáme lineární regresí logaritmu naměřených dat, viz tabulka 3.1.

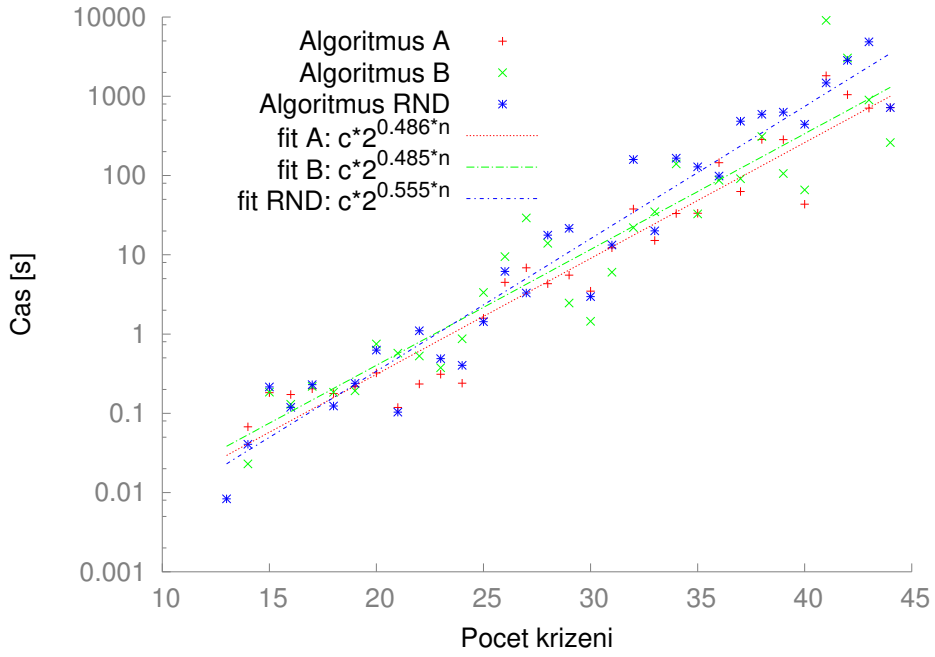
Podle výsledků Jonesův polynom nejrychleji počítá algoritmus A, který volí křížení ve stěně ohraničené dvěma hranami. Odhad jeho průměrné časové složi-



Obrázek 3.4: Graf dob běhu algoritmů na náhodných uzlech proložené křivkami, logaritmická škála.



Obrázek 3.5: Graf dob běhu algoritmů na náhodných alternujících uzlech proložené křivkami, logaritmická škála.



Obrázek 3.6: Graf dob běhu algoritmů na náhodných lincích proložené křivkami, logaritmická škála.

	A	chyba A	B	chyba B	RND	chyba RND
Uzly	0,487	± 0,020	0,503	± 0,036	0,520	± 0,024
Alternující uzly	0,468	± 0,030	0,519	± 0,033	0,509	± 0,036
Linky	0,486	± 0,024	0,485	± 0,032	0,555	± 0,024

Tabulka 3.1: Odhady koeficientu k průměrné časové složitosti $\mathcal{O}\left(2^{kn+o(n)}\right)$ jednotlivých algoritmů podle druhu dat.

tosti na uzlech a lincích je $\mathcal{O}\left(2^{0,49n+o(n)}\right)$, na alternujících uzlech $\mathcal{O}\left(2^{0.468n+o(n)}\right)$. Volba tohoto křížení tedy v průměru vede k většímu rozmotávání než volba křížení ve stěně typu B nebo náhodná volba křížení.

Algoritmus B dosahuje různých výsledků v závislosti na druhu dat. Na lincích má podobnou průměrnou časovou složitost jako algoritmus A, ovšem na alternujících uzlech odhaduje jeho průměrnou složitost $\mathcal{O}\left(2^{0,519n+o(n)}\right)$, což je ze všech variant nejvíce. Odpovídá to i výsledku testování na tabulkových uzlech, kde byl tento algoritmus také nejpomalejší, neboť uzel bývá zaznamenán v alternujícím nakreslení, pokud takové nakreslení existuje. Ačkoli tedy algoritmus B zaručuje ze všech algoritmů u synů největší rozmotání, viz sekce 2.2.5, může na určitých datech běžet nejpomaleji. Na obecných uzlech ovšem dostáváme podobný odhad průměrné složitosti jako u algoritmu A.

Také výsledky algoritmu RND se liší podle druhu testovaných dat. Nejlépe se mu daří na alternujících uzlech, u nichž průměrnou složitost odhadujeme $\mathcal{O}\left(2^{0,509n+o(n)}\right)$. Naopak nejpomalejší je na lincích s odhadem $\mathcal{O}\left(2^{0,555n+o(n)}\right)$. Náhodná volba křížení pravděpodobně vede k rozpadání na disjuntkní uzly, což prodlužuje výpočet.

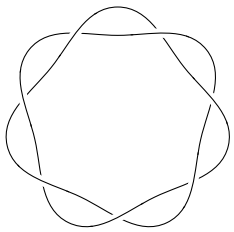
Odhad koeficientu k se často pohybuje kolem hodnoty $k = 0{,}5$. Složitost $\mathcal{O}\left(2^{0,5n}\right)$ odpovídá průběhu algoritmu, v němž dojde u každého syna k rozmotání právě jednoho křížení. Platila by tedy rekurence

$$T(n) = 2T(n-2).$$

3.3 Torusové uzly

Rychlost algoritmů jsme vyzkoušeli na 36 nejmenších torusových uzlech. Jejich PD notaci jsme získali z databáze Knot Atlas [9].

Na grafu dob běhu algoritmů lze rozlišit dva shluky bodů, viz graf 3.8. Spodní shluk odpovídá torusovým uzlům $(2k+1, 2)$, tedy těm, které se obmotají dvakrát podél osy rotace torusu a jejich diagram má tvar dvou zakroucených vláken s $2k+1$ křiženími.



Obrázek 3.7: Torusový uzel (7,2).

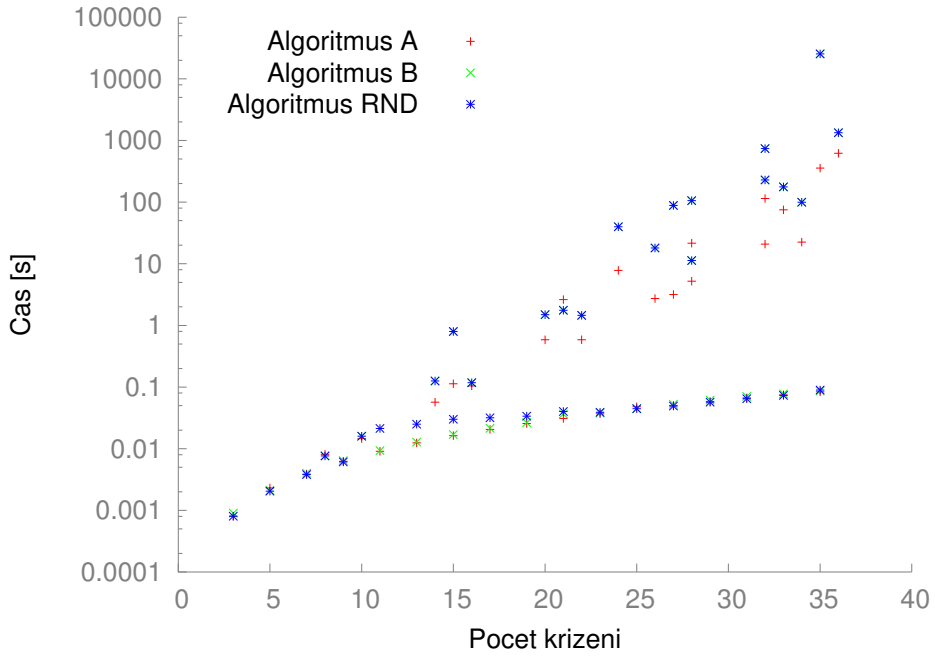
Všechny stěny těchto uzlů jsou ohraničené pouze dvěma hranami. Rozpojením libovolného křížení jedním směrem vznikne torusový link $(2k, 2)$, druhým směrem triviální uzel, který lze rozmotat několika prvními Reidemeistrovými pohyby. Z linku zase rozpojením libovolného křížení vznikne uzel $(2k-1, 2)$. Rozpojení a rozmotávání má lineární časovou složitost, celkově tedy všechny tři algoritmy spočítají Jonesův polynom tohoto typu torusových uzlů v kvadratickém čase, což lze pozorovat na grafu 3.9.

A	chyba A	B	chyba B	RND	chyba RND
0,533	± 0,032	0,638	± 0,048	0,572	± 0,035

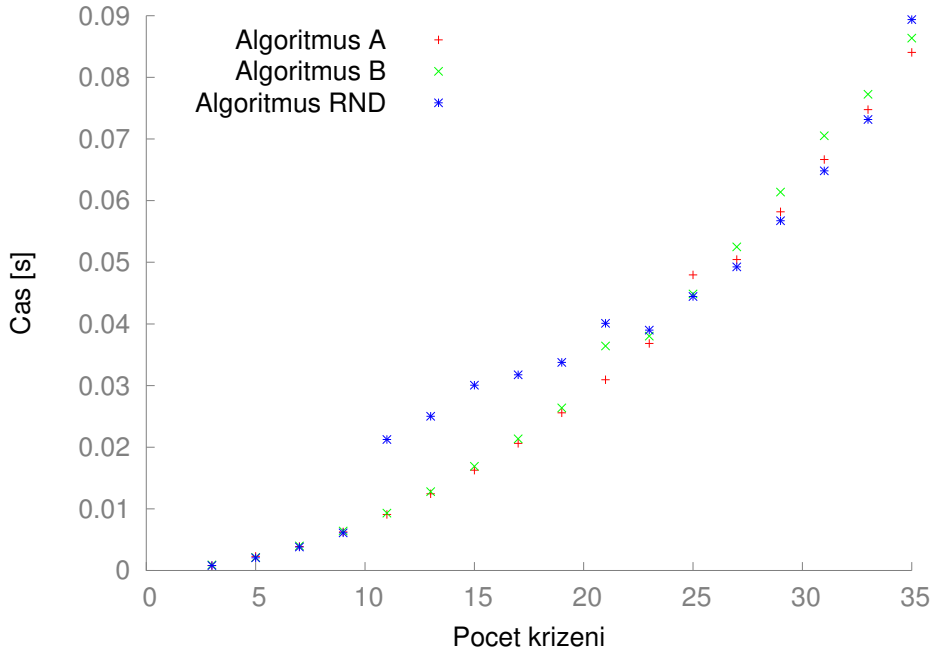
Tabulka 3.2: Odhady koeficientu k průměrné časové složitosti $\mathcal{O}\left(2^{kn} + o(n)\right)$ jednotlivých algoritmů na typu torusových uzlů (p, $q \neq 2$) .

Na torusových uzlech tvaru (p, q), kde $q \neq 2$ již doba běhu algoritmu roste exponenciálně, viz graf 3.10). Opět jsme provedli odhad průměrných časových složitostí, viz tabulka 3.2.

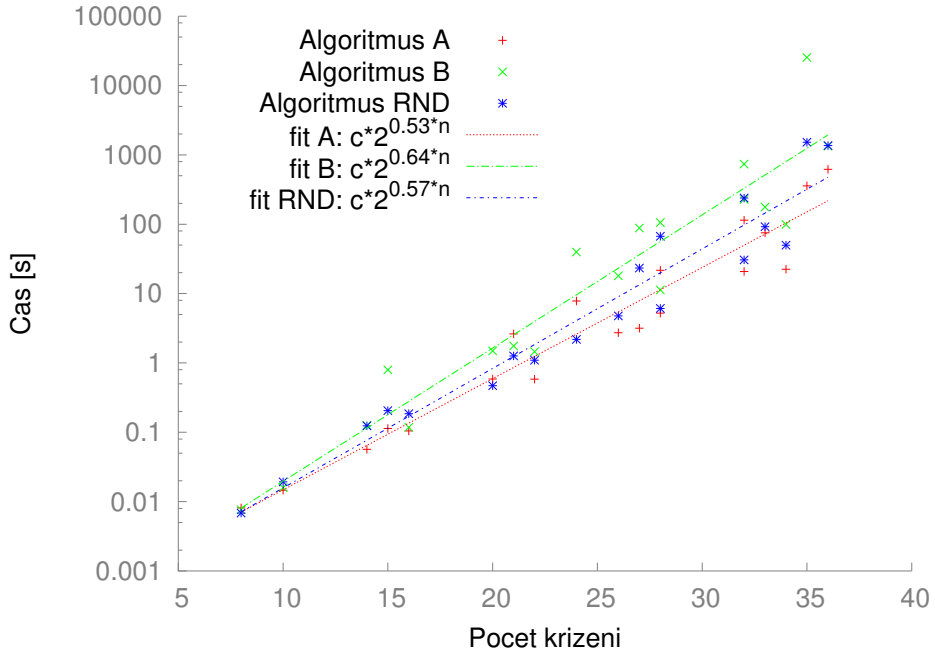
Všechny varianty algoritmů jsou na torusových uzlích pomalejší než na náhodných datech. Nejrychlejší je opět algoritmus A s průměrnou časovou složitostí $\mathcal{O}\left(2^{0,533n+o(n)}\right)$. Naopak nejpomalejší je algoritmus B se složitostí $\mathcal{O}\left(2^{0,638n+o(n)}\right)$, což je nejvyšší složitost, kterou jsme při testování získali.



Obrázek 3.8: Graf dob běhu algoritmů na 36 torusových uzlech, logaritmická škála.



Obrázek 3.9: Doby běhu algoritmů na torusových uzlech (3,2) až (35,2).



Obrázek 3.10: Doby běhu algoritmů na malých torusových uzlech (p, $q \neq 2$) proložené křivkami, logaritmická škála.

Závěr

Hlavním cílem práce bylo odvodit a implementovat algoritmus na výpočet Jonesova polynomu. V první kapitole jsme dokázali souvislost mezi Jonesovým a závorkovým polynomem. Na základě tohoto vztahu jsme v druhé kapitole odvodili algoritmus založený na rekurzivním rozpojování křížení a postupném rozpojování zadaného diagramu. Dokázali jsme, že vhodnou volbou křížení, které rozpojíme, získáme algoritmus s časovou složitostí $\mathcal{O}\left(2^{0,832n}\right)$, kde n značí počet křížení vstupního diagramu. Zároveň jsme nalezli typ diagramu, na němž algoritmus běží v čase $\Omega\left(2^{0,75n}\right)$, a získali jsme tak dolní odhad složitosti algoritmu.

V poslední části práce jsme shrnuli výsledky testování algoritmu na datech. Algoritmus jsme otestovali na tabulkových uzlech, torusových uzlech a různých větších náhodných lincích. Náhodné linky a uzly jsme vygenerovali pomocí převodu z rovinného grafu. U nejrychlejší varianty algoritmu jsme na náhodných uzlech odhadli průměrnou časovou složitost výpočtu $\mathcal{O}\left(2^{0,468n+o(n)}\right)$. Taktéž jsme zjistili, že na jednoduchém typu torusových uzlů běží algoritmus v kvadratickém čase. Na složitějších torusových uzlech dosahuje algoritmus horších výsledků, průměrnou složitost výpočtu Jonesova polynomu torusových uzlů jsme odhadli $\mathcal{O}\left(2^{0,533n+o(n)}\right)$.

Seznam použité literatury

[1] Vaughan F. R. Jones. A polynomial invariant for knots via von Neumann algebras. *Bull. Amer. Math. Soc. (N.S.)*, 12(1):103–111, 1985.

[2] Peter Cromwell. *Knots and links*. Cambridge University Press, Cambridge, UK New York, 2004. ISBN 978-0521839471.

[3] Colin Adams. *The knot book : an elementary introduction to the mathematical theory of knots*. American Mathematical Society, Providence, R.I, 2004. ISBN 978-0821836781.

[4] Vaughan F. R. Jones. The Jones Polynomial. `https://math.berkeley.edu/~vfr/jones.pdf`, 2005.

[5] F. Jaeger, D. L. Vertigan„ D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108(1):35–53, 1990.

[6] Scott Aaronson et al. Complexity Zoo: Sharp-P. `https://complexityzoo.uwaterloo.ca/Complexity_Zoo:Symbols#sharpp`. [Online; navštíveno dne 10. 7. 2018].

[7] Jiří Matoušek, Nešetřil Jaroslav. *Kapitoly z diskrétni matematiky*. Karolinum, Praha, 2009. ISBN 978-80-246-1740-4.

[8] J. C. Cha, C. Livingston. KnotInfo: Table of Knot Invariants. `http://www.indiana.edu/~knotinfo`. [Online; navštíveno dne 2. 7. 2018].

[9] Dror Bar-Natan, Scott Morrison, et al. The Knot Atlas. `http://katlas.org`. [Online; navštíveno dne 2. 7. 2018].

Seznam obrázků

Seznam tabulek

Seznam algoritmů