VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Komponenta výukového serveru TI - P-úplné problémy

Component of Teaching Server for Theoretical Computer Science - Pcomplete problems

Tomáš Kirnig

Bakalářská práce

Vedoucí práce: Ing. Martin Kot, Ph.D.

Ostrava, 2025

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/24/132422
vsboee2403DA70

Zadání bakalářské práce

Student: **Tomáš Kirnig**

Studijní program: B0613A140014 Informatika

Téma: Komponenta výukového serveru TI - P-úplné problémy

Component of Teaching Server for Theoretical Computer Science - P-complete problems

Jazyk vypracování: čeština

Zásady pro vypracování:

V rámci diplomových a bakalářských prací vzniká výukový server pro předměty teoretické informatiky. Jedná se o sadu dynamických webových stránek umožňujících studentům pochopení různých typů úloh a problémů. Na rozdíl od běžných výukových textů s pevně daným počtem ukázkových příkladů umí tyto stránky generovat libovolně mnoho ukázek na základě vstupů od uživatele. Cílem této konkrétní bakalářské práce je vytvořit komponentu pro pomoc s výukou tzv. P-úplných problémů.

Vytvořte dynamické webové stránky umožňující uživateli následující:
1. Simulovat výpočet řešení problému Monotone Circuit Value Problem (MCVP) a alespoň 2 dalších P-úplných problémů.
2. Vstupy těchto algoritmů bude moci uživatel zadávat třemi způsoby:
a) Vhodným, uživatelsky přívětivým, způsobem ručně.
b) Nechat si vstup vygenerovat zcela náhodně podle nastavených parametrů.
c) Vybrat z předpřipravené sady vhodně zvolených vstupů.
3. Bude možné si zobrazit převod instance problému MCVP na ty dva zvolené P-úplné problémy. Přitom:
a) Instanci MCVP pro převod bude možné zadat kterýmkoliv z výše uvedených způsobů.
b) Převod si bude moci uživatel krokovat se zobrazením slovního vysvětlení jednotlivých kroků.
c) Na převodem vytvořenou instanci bude opět možné použít výše požadovanou simulaci výpočtu řešení.

Seznam doporučené odborné literatury:

[1] Miyano, S., Shiraishi, S., Shoudai, T.: "A List of P-Complete Problems", Kyushu University, RIFIS-TR-CS-17, December 1990, dostupné z URL: https://catalog.lib.kyushu-u.ac.jp/opac_download_md/3123/rifis-tr-17.pdf
[2] Sawa, Z.: "Teoretická informatika", podklady pro přednášky, VŠB - Technická univerzita Ostrava, dostupné z URL: https://www.cs.vsb.cz/sawa/ti/slides/ti-slides-03.pdf
[3] Papadimitriou, C.: Computational Complexity, Addison Wesley, 1993
[4] Arora, S., Barak, B.: Computational Complexity: A Modern Approach, Cambridge University Press, 2009

17. listopadu 2172/15
708 00 Ostrava-Poruba
Česká republika

spojovatelka: +420 597 321 111
epodatelna: epodatelna@vsb.cz
ID datové schránky: d3kj88v

IČ: 61989100
DIČ: CZ61989100

email: studijni.fei@vsb.cz
www.fei.vsb.cz

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Kot, Ph.D.**

Datum zadání: 01.09.2024

Datum odevzdání: 30.04.2025

Garant studijního programu: doc. Mgr. Miloš Kudělka, Ph.D.

V IS EDISON zadáno: 30.10.2024 09:50:55

Abstrakt

Tato bakalářská práce se zabývá vývojem výukové webové aplikace pro demonstraci P-úplných problémů. Cílem je usnadnit studentům pochopení a procvičování těchto problémů. Aplikace se zaměřuje na tři P-úplné problémy: *Monotone Circuit Value Problem* (MCVP), *prázdnou gramatiku* a *kombinatorickou hru*. Práce zahrnuje implementaci modulů pro interaktivní zadávání vstupů (ručně, náhodnou generací nebo výběrem z připravené sady) a simulaci jejich řešení. Uživatel může také sledovat krokový převod instance MCVP na další zmíněné problémy a následně jejich řešení.

Klíčová slova

Teoretická informatika, P-úplné problémy, Monotone Circuit Value Problem, webová aplikace, simulace, převod instancí

Abstract

This bachelor's thesis focuses on the development of a teaching-oriented web application for illustrating P-complete problems. The main goal is to facilitate students' understanding and practice of such tasks. The application focuses on the *Monotone Circuit Value Problem* (MCVP), *Empty Grammar*, and the *Combinatorial Game*. The project implements modules for interactive input of problem instances (manually, via random generation, or by selecting from a pre-defined set) and provides a simulation of their solutions. Users can also observe a step-by-step reduction from an MCVP instance to the other mentioned problems and subsequently explore how those are solved. The resulting application demonstrates key concepts of theoretical computer science, including the notion of P-completeness, and provides a flexible basis for educational use.

Keywords

Theoretical computer science, P-complete problems, Monotone Circuit Value Problem, web application, simulation, instance reduction

Obsah

Seznam použitých zkratek a symbolů

MCVP	–	Monotone Circuit Value Problem
P	–	Třída problémů řešitelných v polynomiálním čase
NC	–	Nick's Class – třída efektivně paralelizovatelných problémů

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Teoretická informatika poskytuje způsob jak zkoumat, porozumět a optimalizovat možnosti a limity výpočetních systémů. Hlavním bodem tohoto zkoumání je teorie složitosti, která klasifikuje problémy na základě zdrojů potřebných pro jejich vyřešení, jako je čas a paměť. Jednou z nejvýznamnějších složitostních tříd je třída P, zahrnující problémy řešitelné v polynomiálním čase – tedy takové, jejichž časová složitost lze vyjádřit jako $O(n^k)$ pro nějakou konstantu k (například $k = 2$ pro kvadratickou složitost, $k = 3$ pro kubickou), kde n je velikost vstupu – na deterministickém Turingově stroji [1]. Přestože jsou problémy v této třídě obvykle pokládány za „efektivně řešitelné“, projevují se mezi nimi významné odlišnosti, obvzvláště když začneme řešit jejich paralelizovatelnost.

V tomto kontextu hraje klíčovou roli podmnožina *P-úplných* problémů (*P-complete* problems). P-úplné problémy představují výpočetně nejnáročnější úlohy v rámci třídy P. Jsou to problémy, na které lze s logaritmickou paměťovou složitostí převést jakýkoliv jiný problém z třídy P. Hlavní problém u této podmnožiny problémů spočívá v předpokladu, že P-úplné problémy pravděpodobně nelze efektivně paralelizovat. To znamená, že tyto úlohy nespadají do třídy NC, která obsahuje problémy řešitelné v polylogaritmickém čase $O(\log^k n)$ pomocí paralelního výpočtu s polynomiálně mnoha procesory [2]. Z toho vyplývá, že na rozdíl od NC problémů řešení P-úplných problémů vyžaduje sekvenční zpracování. Studium P-úplnosti nám tak pomáhá vymezit hranici mezi paralelizovatelným a čistě sekvenčním výpočtem.

Tento projekt představuje interaktivní ukázku konceptu P-úplnosti prostřednictvím webové aplikace. Cílem je vytvořit nástroj, který umožní uživatelům vizualizovat a pochopit výpočet a převod mezi zvolenými P-úplnými problémy.

Jako hlavní problém byl zvolen *Monotone Circuit Value Problem* (MCVP), ve kterém se vyhodnocuje logický obvod složený z pouze dvou logických hradel [3] (tedy logický součin a součet). Pro demonstraci univerzálnosti konceptu P-úplnosti aplikace implementuje také simulace dvou dalších problémů:

- **Kombinatorické hry na grafu:** Úloha analyzující existenci vítězné strategie v deterministické hře dvou hráčů [3].

- **Prázdnost bezkontextových gramatik:** Problém rozhodující, zda daná gramatika generuje neprázdný jazyk [3].

Hlavním přínosem vytvořené aplikace je možnost vizualizace nejen samotného řešení těchto úloh, ale především *převodů* (redukcí) mezi nimi. Uživatel může sledovat krokovou transformaci instance MCVP na instanci hry nebo gramatiky, což názorně ilustruje princip polynomiálních redukcí a vzájemnou převoditelnost P-úplných problémů [2].

Výsledkem práce je webová aplikace navržená jako flexibilní nástroj pro výuku. Umožňuje uživatelům pracovat s vlastními vstupy, generovat náhodné instance pro testování a využívat předpřipravené sady úloh.

Text práce je členěn do logických celků. Po úvodu následuje kapitola věnovaná použitým technologiím a architektuře aplikace. Jádro práce tvoří tři kapitoly, z nichž každá se detailně věnuje jednomu z implementovaných problémů: nejprve Monotone Circuit Value Problem (MCVP), následně kombinatorické hry na grafu a nakonec problém prázdnosti bezkontextových gramatik. Závěr práce shrnuje dosažené výsledky a navrhuje možnosti dalšího rozšíření.

Kapitola 2

Použité technologie a architektura

Tato kapitola popisuje technologický základ vytvořené aplikace a její architekturu. Výběr technologií byl veden požadavky na moderní, interaktivní a snadno rozšiřitelné řešení vhodné pro výukové účely.

2.1 Webové technologie

Pro implementaci výukové aplikace jsme zvolili webové technologie, které nabízejí řadu výhod oproti desktopovým nebo mobilním aplikacím. Webová aplikace nevyžaduje instalaci a běží v libovolném moderním webovém prohlížeči, což zajišťuje maximální dostupnost pro uživatele napříč různými platformami a operačními systémy. Dalším přínosem je snadná údržba – aktualizace aplikace se projeví okamžitě u všech uživatelů bez nutnosti distribuce nových verzí.

2.2 React a Vite

Jako hlavní framework pro vývoj uživatelského rozhraní byl zvolen *React* [4]. React je moderní JavaScriptová knihovna vyvinutá společností Meta (dříve Facebook), která umožňuje vytvářet interaktivní uživatelská rozhraní na bázi komponent. Hlavní výhodou Reactu je koncept *reaktivity* – uživatelské rozhraní se automaticky aktualizuje při změně dat bez nutnosti manuální manipulace s DOM (Document Object Model).

React využívá deklarativní přístup k tvorbě UI. Na rozdíl od tradičního imperativního programování, kde vývojář musí krok za krokem instruovat prohlížeč, jak upravit rozhraní, v Reactu stačí popsat, jak má výsledné rozhraní vypadat, a React se postará o potřebné změny v DOM. Tento přístup výrazně zjednodušuje vývoj komplexnějších aplikací a minimalizuje chyby spojené s nekonzistentním stavem.

Pro sestavení a vývoj aplikace jsme použili *Vite* [5]. Vite je moderní *bundler* – tedy nástroj, který spojuje všechny zdrojové soubory aplikace (JavaScript, CSS, obrázky) do optimalizované podoby připravené pro odeslání do prohlížeče. Během vývoje Vite nabízí významnou výhodu: místo zdlou-

havého spojování všech souborů do jednoho velkého balíku servíruje jednotlivé moduly přímo do prohlížeče v jejich původní podobě. To výrazně urychluje spouštění vývojového serveru a umožňuje okamžité promítnutí změn v kódu (tzv. hot module replacement). Pro produkční nasazení pak Vite vytvoří klasický optimalizovaný build, kde jsou všechny soubory sloučeny, zmenšeny a připraveny pro rychlé načítání.

2.3 Správa stavu a komponentová architektura

Aplikace je strukturována jako *Single Page Application* (SPA), kde celá aplikace běží v rámci jedné HTML stránky a navigace mezi jednotlivými moduly probíhá bez opětovného načítání stránky. Hlavní komponenta `App.jsx` funguje jako kořen aplikační struktury a spravuje globální stav aplikace pomocí React Hooks, především `useState` pro udržení aktuální stránky a předávaných dat.

Architektura je založena na principu *unidirectional data flow* (jednosměrný tok dat), kdy data tečou z rodičovské komponenty do potomků prostřednictvím properties (props), zatímco akce a události se propagují zpět nahoru pomocí callback funkcí. Tento přístup zajišťuje předvídatelné chování aplikace a usnadňuje ladění.

Aplikace je rozdělena do logických modulů podle jednotlivých problémů (viz kapitoly 3, 4 a 5). Každý modul je organizován do vlastní složky obsahující:

- **Hlavní komponentu** – kontejnerovou komponentu řídící celý modul
- **Utils** – pomocné funkce obsahující algoritmy (parsery, generátory, evaluátory)
- **InputSelectionComponents** – komponenty pro různé způsoby zadání vstupu
- **Vizualizační komponenty** – komponenty pro grafické zobrazení problémů a jejich řešení

Pro sdílení společné funkcionality mezi moduly byly vytvořeny *custom hooks* (vlastní React Hooks) v adresáři `src/Hooks`. Hook `useGraphColors` centralizuje správu barev používaných ve vizualizacích a načítá je z CSS proměnných, což umožňuje snadnou změnu barevného schématu. Hook `useGraphSettings` poskytuje konfigurační parametry pro force-directed grafy, jako jsou poloměry uzlů, síly odpudivosti a vzdálenosti hran.

2.4 Vizualizace grafů

Pro vizualizaci stromových a grafových struktur používá aplikace knihovnu *react-force-graph-2d* [6], která je React wrapperem nad výkonnou knihovnou *D3.js* (Data-Driven Documents) [7]. D3.js je de facto standardem pro tvorbu datových vizualizací ve webovém prostředí a poskytuje rozsáhlé možnosti pro manipulaci s DOM na základě dat.

2.4.1 Force-directed layout

Klíčovým konceptem využívaným pro rozmístění uzlů v grafu je *force-directed layout* (rozložení řízené silami). Tento algoritmus modeluje graf jako fyzikální systém, kde uzly představují nabitá tělesa odpudící se navzájem a hrany fungují jako pružiny přitahující spojené uzly k sobě. Simulace běží iterativně, dokud systém nedosáhne stavu s minimální energií, tedy vizuálně příjemného a čitelného rozložení.

Konkrétně aplikace využívá následující síly:

- **Link force** – síla pružin mezi spojenými uzly, udržuje požadovanou vzdálenost hran
- **Charge force** – odpudivá síla mezi všemi uzly, zabraňuje jejich překrývání
- **Collision force** – detekuje a řeší kolize mezi uzly na základě jejich poloměrů
- **Center force** – udržuje celý graf vycentrovaný v zobrazovací ploše

Pro stromové struktury v MCVP modulu (viz kapitola 3) je aktivován speciální *DAG mód* (Directed Acyclic Graph), který automaticky organizuje uzly do hierarchických úrovní shora dolů (`dagMode="td"`). To zajišťuje, že výraz je vizualizován tak, jak je běžné v teoretické informatice – kořen nahoře, listy dole.

2.4.2 Vlastní vykreslování

Knihovna react-force-graph-2d umožňuje definovat vlastní vykreslovací funkce (`nodeCanvasObject`, `linkCanvasObjectMode`), což aplikace využívá pro vizualizaci stavů uzlů a hran během evaluace. Například v MCVP modulu se uzly vybarvují podle jejich typu (proměnné vs. operátory) a vyhodnoceného stavu (0 nebo 1), zatímco hrany mění barvu podle toho, zda již byly zpracovány v průchodu stromem.

Vykreslování probíhá na HTML5 Canvas elementu, který nabízí vysoký výkon i pro grafy s desítkami či stovkami uzlů. Oproti SVG přístupu je Canvas renderování efektivnější pro časté překreslování, což je klíčové pro animace krokování algoritmy.

2.5 Stylování a responzivita

Pro stylování uživatelského rozhraní aplikace využívá framework *Bootstrap 5* [8]. Bootstrap poskytuje předpřipravené CSS třídy pro tvorbu responzivního layoutu, komponent jako jsou tlačítka, formuláře, modální okna a navigační prvky. Použití Bootstrapu výrazně urychlilo vývoj a zajistilo konzistentní vzhled napříč celou aplikací.

Responzivita je implementována pomocí Bootstrapového grid systému založeného na Flexboxu. Layout se automaticky přizpůsobuje velikosti obrazovky, přičemž na mobilních zařízeních se navigace

transformuje do off-canvas menu, které se vysouvá ze strany obrazovky. Toto řešení zajišťuje dobrou použitelnost aplikace i na tabletech a smartphonech.

Vedle Bootstrapu jsou specifické styly definovány v centrálním souboru `style.css`, který využívá CSS custom properties (proměnné) pro správu barevného schématu. Tento přístup umožňuje snadnou změnu celého vzhledu aplikace změnou několika proměnných.

2.6 Správa vstupů a výstupů

Každý modul aplikace implementuje tři základní způsoby zadání vstupu:

1. **Manuální zadání** – uživatel zadává vstup prostřednictvím textového pole (např. výraz v notaci pro MCVP) nebo interaktivního grafického editoru

2. **Náhodné generování** – algoritmus generuje náhodnou instanci problému na základě parametrů zadaných uživatelem

3. **Připravené sady** – výběr z předpřipravených příkladů uložených ve formátu JSON v adresáři `Sady/`

Pro import a export dat aplikace používá formát JSON, který je standardním formátem pro výměnu dat ve webových aplikacích. Komponenta `FileTransferControls` (viz oddíl 2.3) poskytuje jednotné rozhraní pro stahování dat jako JSON soubory a jejich načítání pomocí drag-and-drop nebo výběru souboru.

2.7 Notifikace a zpětná vazba

Pro zobrazení upozornění, chybových hlášek a potvrzení úspěšných akcí aplikace využívá knihovnu *react-toastify* [9]. Tato knihovna poskytuje elegantní toast notifikace, které se objevují v rohu obrazovky a automaticky mizí po nastavené době. Notifikace jsou použity zejména při:

- Chybách při parsování vstupů

- Úspěšném importu/exportu dat

- Varováních při nevalidních operacích

Toast notifikace poskytují okamžitou, nenápadnou zpětnou vazbu bez přerušení práce uživatele, což je v souladu s doporučenými praktikami UX designu.

2.8 Struktura kódu a modularita

Zdrojový kód aplikace je organizován v adresáři `src/` následovně:

- **src/** – kořenový adresář obsahující `App.jsx`, `main.jsx` a globální styly
- **src/Components/** – všechny React komponenty rozdělené do podadresářů podle účelu:
  - **MCVP/** – modul pro Monotone Circuit Value Problem
  - **CombinatorialGame/** – modul pro kombinatorické hry
  - **Grammar/** – modul pro bezkontextové gramatiky
  - **Conversions/** – komponenty pro vizualizaci převodů mezi problémy
  - **Common/** – sdílené komponenty (modální okna, tlačítka, file handling)
  - **HPVisual/** – komponenty pro vizualizaci na úvodní stránce
- **src/Hooks/** – custom React hooks pro sdílenou logiku

Tato struktura zajišťuje jasné oddělení odpovědností a usnadňuje orientaci v kódu. Každý modul je relativně nezávislý a může být modifikován nebo rozšiřován bez dopadu na ostatní části aplikace.

2.9 Vývojové nástroje a kontrola kvality

Pro zajištění kvality kódu aplikace využívá *ESLint* [10] – nástroj pro statickou analýzu JavaScriptového kódu. ESLint kontroluje dodržování kódovacích standardů, detekuje potenciální chyby a anti-patterny. Konfigurační soubor `eslint.config.js` obsahuje pravidla specifická pro React aplikace, včetně doporučení pro používání Hooks a správu závislostí.

Během vývoje je využíván Vite development server s podporou Hot Module Replacement, který umožňuje okamžité promítnutí změn v kódu do běžící aplikace bez nutnosti manuálního obnovení stránky. To výrazně zrychluje vývojový cyklus a umožňuje rychlejší iterace.

2.10 Nasazení a distribuce

Pro produkční nasazení aplikace Vite vytvoří optimalizovaný build pomocí příkazu `npm run build`. Výsledné soubory v adresáři `dist/` obsahují minifikovaný JavaScript, CSS a další statické assety. Tyto soubory mohou být nahrány na libovolný webový server nebo hosting platformu podporující statické webové stránky.

Vzhledem k tomu, že aplikace je čistě client-side (veškerá logika běží v prohlížeči uživatele), nevyžaduje serverovou infrastrukturu pro zpracování požadavků. To výrazně zjednodušuje nasazení a hosting, který může být realizován například pomocí služeb GitHub Pages, Netlify, Vercel nebo klasického webového hostingu.

Kapitola 3

Monotone Circuit Value Problem

Kapitola 4

Kombinatorické hry na grafu

Kapitola 5

Prázdnost bezkontextových gramatik

Kapitola 6

Závěr

Literatura

1. PAPADIMITRIOU, Christos H. *Computational Complexity*. Addison-Wesley, 1993. ISBN 978-0201530827.
2. ARORA, Sanjeev; BARAK, Boaz. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. ISBN 978-0521424264.
3. SAWA, Zbyněk. *Teoretická informatika – Podklady pro přednášky* [Online]. 2024. Dostupné také z: `https://www.cs.vsb.cz/sawa/ti/slides/ti-slides-08.pdf`.
4. META PLATFORMS, INC. *React – A JavaScript library for building user interfaces* [Online]. 2024. Dostupné také z: `https://react.dev/`.
5. VITE TEAM. *Vite – Next Generation Frontend Tooling* [Online]. 2024. Dostupné také z: `https://vitejs.dev/`.
6. ASTURIANO, Vasco. *react-force-graph – React component for 2D, 3D, VR and AR force directed graphs* [Online]. 2024. Dostupné také z: `https://github.com/vasturiano/react-force-graph`.
7. BOSTOCK, Mike. *D3.js – Data-Driven Documents* [Online]. 2024. Dostupné také z: `https://d3js.org/`.
8. BOOTSTRAP TEAM. *Bootstrap – The most popular HTML, CSS, and JS library in the world* [Online]. 2024. Dostupné také z: `https://getbootstrap.com/`.
9. KHADRA, Fadi. *react-toastify – React notification made easy* [Online]. 2024. Dostupné také z: `https://fkhadra.github.io/react-toastify/`.
10. ESLINT TEAM. *ESLint – Pluggable JavaScript linter* [Online]. 2024. Dostupné také z: `https://eslint.org/`.

Příloha A

Dlouhý zdrojový kód