ŽILINSKÁ UNIVERZITA V ŽILINE

AUTOREFERÁT
DIZERTAČNEJ PRÁCE

Žilina, máj 2020

Mgr. Adam Dudáš

Žilinská univerzita v Žiline
Fakulta riadenia a informatiky

Mgr. Adam Dudáš

Autoreferát dizertačnej práce

Optimalizácia dekompozície paralelných a distribuovaných výpočtov vybraných grafových úloh vo vysokovýkonnom počítaní

na získanie akademického titulu **„philosophiae doctor“ (v skratke PhD.)**
v študijnom programe doktorandského štúdia

aplikovaná informatika

v študijnom odbore:
informatika

Žilina, máj 2020

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia **na katedre informatiky, Fakulte riadenia a informatiky Žilinskej univerzity v Žiline**

Predkladateľ: **Mgr. Adam Dudáš**
Katedra informatiky
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Školiteľ: **doc. Ing. Jarmila Škrinárová, PhD.**
Katedra informatiky
Fakulta prírodných vied
Univerzita Mateja Bela

Oponent: **doc. Ing. Penka Martincová, PhD.**
Katedra mediamatiky a kultúrneho dedičstva
Fakulta humanitných vied
Žilinská univerzita v Žiline

Oponent: **prof. RNDr. Valerie Novitzká, PhD.**
Katedra počítačov a informatiky
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach

Autoreferát bol rozoslaný dňa: .....

Obhajoba dizertačnej práce sa koná dňa 18.8.2020 o 12:30 h. pred komisiou pre obhajobu dizertačnej práce schválenou pracovnou skupinou odborovej komisie v študijnom odbore **informatika v študijnom programe aplikovaná informatika**, vymenovanou dekanom Fakulty riadenia a informatiky Žilinskej univerzity v Žiline dňa .....

prof. Ing. Karol Matiaško, PhD.
predseda pracovnej skupiny odborovej komisie
študijného programu **aplikovaná informatika**
v študijnom odbore **informatika**

Fakulta riadenia a informatiky
Žilinská univerzita
Univerzitná 8215/1
010 26 Žilina

ABSTRAKT

DUDÁŠ, Adam. Optimalizácia dekompozície paralelných a distribuovaných výpočtov vybraných grafových úloh vo vysokovýkonnom počítaní [dizertačná práca]. Žilinská univerzita v Žiline. Fakulta riadenia a informatiky. Katedra informatiky. Školiteľ: doc. Ing. Jarmila Škrinárová, PhD. Žilina, 2020.

V tejto dizertačnej práci sa zameriavame na prezentovanie analyzovaných poznatkov, algoritmov, metodík a nástrojov relevantných pri riešení problematiky optimalizácie dekompozície paralelných a distribuovaných výpočtov vybraných grafových úloh vo vysokovýkonnom počítaní. Ako prostriedok riešenia uvedenej problematiky sme zvolili problém hranového trojofarbovania grafov, konkrétne sa sústredíme na regulárne hranové trojofarbenie pri skupine kubických grafov nazývaných snarky. Práca obsahuje analýzu teoretických poznatkov týkajúcich sa paralelných a distribuovaných výpočtov, dekompozície výpočtov a vybraných poznatkov z oblasti teórie grafov. Analyzujeme aktuálny stav riešenia z pohľadu paralelných a distribuovaných systémov a z pohľadu algoritmov využívaných pri ofarbovaní grafov. Prezentujeme navrhnuté metodiky, nástroje a algoritmy, ktoré boli experimentálne overené. Všetky experimenty vyhodnocujeme pomocou vybraných kritérií.

Kľúčové slová: paralelné počítanie, distribuované počítanie, dekompozícia výpočtu, vysokovýkonné počítanie, ofarbovanie grafov, snark

OBSAH

ÚVOD

Teória grafov v sebe zahŕňa mnohé problémy, ktoré je možné riešiť pomocou operácií na grafoch. V našej práci sa zaoberáme **hranovým ofarbovaním grafov**, ktoré je relevantné v niekoľkých oblastiach – rozvrhovanie, prideľovanie rádiových frekvencií, optimalizácia kompilátorov alebo SAT solvery.

Keďže hranové ofarbovanie grafov je **NP-úplný problém**, používanie paralelných a distribuovaných metód je na jeho výpočet žiadané. V našom prípade počítame s potrebou ofarbenia veľkého množstva grafov. Túto úlohu **dekomponujeme** na menšie podúlohy, ktoré riešime paralelne alebo distribuovane.

Naším cieľom je vytvoriť metódy a algoritmy na efektívne paralelné, distribuované alebo kombinované ofarbenie veľkých množín grafov pričom dbáme na optimalizačné kritériá ako **čas ukončenia poslednej úlohy** a tak isto na **dĺžku trvania výpočtu jednotlivých podúloh**.

V prvej časti práce uvádzame pojmy z oblasti **paralelného a distribuovaného počítania**, ktoré sú relevantné v našej práci.

Druhá kapitola práce sa vzťahuje na stručné predstavenie pojmov a konceptov z problematiky teórie grafov, ktoré riešime v tejto práci.

V tretej kapitole prezentujeme dosiahnuté výsledky. Prezentujeme **vlastné algoritmy a metódy** na hranové ofarbovanie veľkých množín grafov s využitím paralelného a distribuovaného počítania.

Na záver práce uvádzame **vyhodnotenie dosiahnutých výsledkov** a ponúkame pohľad na ďalšie možné oblasti, ktoré je potrebné v budúcnosti skúmať.

CIELE PRÁCE

Hlavným cieľom práce je návrh, implementácia a overenie algoritmov, metodík a nástrojov, ktoré prispejú k optimalizácii dekompozície paralelných aplikácií pri práci s vybranými ťažkými úlohami z oblasti teórie grafov – v našom prípade ide konkrétne o hranové trojofarbovanie grafov. Konkrétnymi cieľmi práce je potvrdenie alebo zamietnutie nasledujúcich hypotéz:

- H1 - *Algoritmus hranového backtrackingu nie je citlivý na počiatočný vrchol ofarbovania grafu. To znamená, že ak pri výpočte použijeme rôznu postupnosť hrán, bude doba výpočtu algoritmu pre zadaný graf rovnaká.*
- H2 - *Neexistuje také poradie ofarbovania hrán grafu, že po jeho aplikovaní na množinu grafov, bude čas ofarbenia celej množiny grafov redukovaný. Táto hypotéza súvisí s optimalizáciou časovej náročnosti algoritmu hranového backtrackingu.*
- H3 - *Nie je možné rozdeliť množinu grafov na podmnožiny (zhluky) tak, že pre každú podmnožinu grafov je možné nájsť také poradie ofarbovania hrán, ktoré znižuje čas potrebný na ofarbenie každého grafu v danej podmnožine*.
- H4 - *Neexistuje také poradie ofarbovania hrán grafu, že čas hranového ofarbovania je pre každý graf z množiny rovnaký (resp. je v stanovenej tolerancii - intervale). Táto hypotéza súvisí so správnou dekompozíciou úlohy podľa 1.3.*

Okrem cieľov práce formalizovaných v hypotézach je potrebné preskúmať možnosti paralelizácie výpočtov týkajúcich sa hranového trojofarbovania grafov. Tento cieľ je výsledkom osobnej komunikácie s hlavným riešiteľom v rámci projektu VEGA 1/0487/17 - Algoritmy na grafoch a algebraických štruktúrach a tiež bol zahrnutý v slovenskom manifeste týkajúcom sa smerovania výskumu v oblasti teórie grafov z roku 2015.

1 PARALELNÉ A DISTRIBUOVANÉ POČÍTANIE

Paralelné a distribuované počítanie je prítomné v niekoľkých odvetviach informatiky – algoritmy, počítačové architektúry, siete, operačné systémy alebo softvérové inžinierstvo. Paralelné a distribuované počítanie stavia na základných systémových konceptoch a pridáva k nim niekoľko konceptov ako súbežné vykonávanie procesov, vzájomné vylúčenie, zasielanie správ a modely so zdieľanou pamäťou.

V práci sa venujeme výpočtu ofarbenia veľkých množín grafov pomocou paralelných, distribuovaných alebo kombinovaných algoritmov. Zameriavame sa na dekompozíciu daného problému a hľadáme taký model, ktorý uľahčí a zrýchli výpočet.

1.1 PARALELNÉ A DITRIBUOVANÉ POČÍTANIE

Paralelné výpočty môžeme definovať ako výpočty, ktoré simultánne používajú viaceré výpočtové prostriedky. Pre svoju aktivitu využívajú viaceré procesory (angl. Processing element, PE). Ich úlohy sú delené na diskrétne časti, zvané podúlohy, ktoré je možné riešiť súbežne a každá z týchto diskrétnych častí je následne rozdelená na postupnosť inštrukcií, ktoré je možné vykonávať súbežne na rôznych procesoroch [1].

Komponenty distribuovaného systému sú počítače, navzájom prepojené sieťou, ktoré komunikujú a koordinujú svoju prácu zasielaním správ [2]. Sieťou prepojené počítače môžu byť fyzicky rozdelené akoukoľvek vzdialenosťou – môžu byť na rôznych kontinentoch, v rovnakej budove ale aj v rovnakej miestnosti. Definícia distribuovaných systémov v sebe zahŕňa niekoľko významných konceptov [2], najmä súbežnosť, neexistenciu globálnych hodín a nezávislé zlyhania.

1.3 DEKOMPOZÍCIA ÚLOHY

Dekompozíciu problému môžeme opísať ako rozkladanie výpočtov na menšie časti. Pre takéto rozdelenie úlohy je vhodné používať niektoré z **modelov dekompozície**. **Optimalizáciou týchto modelov** myslíme ich využívanie a prípadné upravovanie tak, aby sme minimalizovali čas výpočtu. Na základe dekompozície problému vznikne množina nezávislých alebo vzájomne komunikujúcich procesov (úloh), ktorú možno priradiť vhodnej paralelnej architektúre [1]. Pri prideľovaní úloh na jednotlivé procesory dbáme na vyrovnávanie záťaže s cieľom minimalizovať náklady.

Základom pre dekompozíciu problémov je poznanie **idealizovaných typoch paralelizmov** – jednoduchý, prúdový, expanzívny a masívny paralelizmus.

Pre rozdelenie úlohy na časti je nutné používať niektoré z **modelov dekompozície**. Podľa [1] rozoznávame 4 druhy dekompozície problému: Jednoduchá, Funkcionálna, Hierarchická a Údajová dekompozícia.

Zjednocujúcim podstatným konceptom pri dekomponovaní úloh na podúlohy je **dĺžka trvania jednotlivých podúloh**. V ideálnom prípade by malo vykonanie všetkých podúloh trvať **rovnaký čas**, čo vedie k stavu, keď podúlohy nemusia v systéme čakať na dokončenie dlhšej časti výpočtu.

Autori Grama, Gupta et al. v [3] uvádzajú štyri fundamentálne skupiny techník dekompozície úlohy na podúlohy: Rekurzívna, Dátová, Dekompozícia bádaním (z angl. exploratory decomposition) a Dekompozícia domnienkou (z angl. Speculative decomposition).

Tieto modely sa navzájom nevylučujú – často sú kombinované do **hybridných dekompozičných metód** [3]. Ak je výpočet štruktúrovaný v niekoľkých krokoch, je možné rôzne modely dekompozície uplatniť v rôznych krokoch výpočtu.

2 HRANOVÉ TROJOFARBOVANIE SNARKOV

Táto práca je zameraná na hranové ofarbovanie veľkých množín grafov (NP-úplný problém), pričom ide o konkrétny typ grafov, pri ktorých je samotné hranové ofarbovanie problematické - **snarky**. **Graf** G je dvojica množín V a E, pričom prvky množiny E sú dvojprvkovými podmnožinami množiny V [4]:

$$G = (V, E) \text{ pričom } E \subseteq [V]^2 \qquad (2.1)$$

Základným pojmom, pomocou ktorého sme v našej práci vymedzili skupinu grafov, s ktorými budeme ďalej pracovať je **stupeň vrcholu grafu**. Ak je vrchol V stupňa 3, označujeme ho ako **deg(V) = 3**. Tento pojem označuje počet hrán, ktoré do vybraného vrcholu vchádzajú resp. z neho vychádzajú. **Najvyšší** (maximálny) stupeň vrcholu v grafe G označujeme ako $\Delta(G)$.

V tejto práci sa zaoberáme výhradne **kubickými grafmi**. Graf je kubický práve vtedy, keď sú všetky jeho **vrcholy stupňa tri.** Veľké množstvo problémov z oblasti teórie grafov je riešiteľných práve vtedy, keď sú riešiteľné na množine kubických grafov [5].

2.1 REGULÁRNE HRNOVÉ OFARBOVANIE KUBICKÝCH GRAFOV

Hranovým ofarbením grafu myslíme priradenie farieb jednotlivým hranám grafu. Takéto ofarbenie považujeme za regulárne práve vtedy, keď sa v grafe nenachádza žiadny **konflikt**, tj. so žiadnym z vrcholov grafu nesusedia dve alebo viac hrán ofarbených rovnakou farbou.

Najmenší počet farieb použiteľných pri regulárnom hranovom ofarbovaní grafu G označuje **hranový chromatický index grafu $\chi'(G)$** [4]. Pre kubický graf *G* platí, že $3 \leq \chi'(G) \leq 4$.

Poznáme malú množinu kubických grafov, pre ktoré na regulárne hranové ofarbenie potrebujeme použiť štyri farby. Grafy z tejto množiny kubických grafov sa nazývajú **hranovo 3-neofarbiteľné** alebo **snarky** [6]. Najmenší známy graf, patriaci do skupiny snarkov je Petersenov graf.

Táto skupina grafov sa pri samotnom hranovom ofarbovaní správa problematicky. Snarky majú chromatický index $\chi'(G) = 4$. Aby sme však zistili či je graf *G* snarkom, musíme ho hranovo ofarbiť s využitím troch farieb. Z toho vyplýva, že algoritmus, pomocou ktorého chceme graf *G* ofarbiť, musí prejsť všetkými možnosťami hranového trojofarbenia grafu *G*, aby vylúčil možnosť existencie regulárneho hranového trojofarbenia tohto grafu. Až po prejdení všetkých možných hranových ofarbení troma farbami vieme o grafe povedať či je alebo nie je snarkom.

2.2 ALGORITMUS HRANOVÉHO BACKTRACKINGU

V práci využívame algoritmus hranového backtrackingu, ktorý pracuje na princípe postupného ofarbovania hrán grafu vopred určenou **postupnosťou troch farieb**. Algoritmus sa pri nájdení rozporu v ofarbovaní grafu **vráti na predošlú hranu**, ktorú prefarbí a pokračuje v ďalšom ofarbovaní grafu. Ak ani jedno z možných ofarbení problematickej hrany nie je regulárne, algoritmus sa vracia k hrane **predchádzajúcej obe prefarbované hrany**. Algoritmus pokračuje v tomto postupe pokiaľ nie je celý graf regulárne ofarbený alebo pokiaľ algoritmus neprejde všetkými možnosťami ofarbenia grafu. Časová zložitosť hranového backtrackingu je $O(2^{n-1})$, pričom *n* je počet vrcholov zadaného grafu.

3 MODELY PARALELNÝCH VÝPOČTOV NA OFARBOVANIE GRAFOV

Pri skúmaní vlastností grafov je algoritmus regulárneho hranového ofarbovania grafov len **čiastočnou operáciou**. Preto je potrebné graf ofarbiť v čo najnižšom čase.

V algoritmoch, prezentovaných v tejto kapitole, pracujeme s konceptom **ofarbovania permutovaných kubických grafov**. V aplikáciách využívame programový model vlákien, pričom riešime tvorbu permutácií grafu pomocou konjugácie. Na samotné ofarbovanie jednotlivých grafov používame algoritmus hranového backtrackingu prezentovaný v kapitole 2.2.

V práci sme navrhli 4 hypotézy. Začneme hypotézou H1, ktorá je východisková:

Hypotéza H1. *Algoritmus hranového backtrackingu nie je citlivý na počiatočný vrchol ofarbovania grafu. To znamená, že ak pri výpočte použijeme rôznu postupnosť hrán bude doba výpočtu algoritmu pre zadaný graf rovnaká.*

Nech je graf *G* reprezentovaný maticou susedností. Ak sa potvrdí, že výpočty, ktoré používajú rôzne poradie ofarbovania hrán trvajú rovnako dlho, hypotéza bude potvrdená, v opačnom prípade bude vyvrátená. Aby sme mohli skúmať túto hypotézu, potrebujeme si pre graf *G* vytvoriť množstvo permutácií matice susedností tohto grafu. Takto vytvoríme množstvo modifikácií jedného grafu, ktoré budú mať rôznu postupnosť hrán.

Využijeme preto **izomorfizmus grafov**. Nech graf *G´* a zadaný graf *G* sú izomorfnými grafmi. Dôležitou vlastnosťou izomorfných grafov je, že ľubovoľná dvojica takýchto grafov má rozdielne matice susedností ale zhodné diagramy. V našom prípade to znamená, že ide o zhodné grafy s rozdielnou postupnosťou vrcholov a hrán. Ak je graf *G* reprezentovaný pomocou matice susedností *A*, rôznu postupnosť hrán grafu G (izomorfný graf *G´*) vieme vytvoriť pomocou **permutácie matice susedností grafu** *G*. Permutáciu matice susednosti *A* grafu *G* vypočítame pomocou **konjugácie** (3.1).

$$A' = P^{-1} * A * P \qquad (3.1)$$

kde *A´* je **permutovaná matica susedností grafu *G***, *A* je pôvodná **matica susedností grafu *G***, *P* označuje **permutačnú maticu grafu** (matica naplnená nulami s práve jednou jednotkou v každom riadku a každom stĺpci), P^{-1} označuje **transponovanú permutačnú maticu** *P*.

Takto získaná permutovaná matica susedností A' reprezentuje zmenené poradie hrán pôvodného grafu G. Podľa tohto poradia ofarbujeme hrany grafov v našich experimentoch.

3.1 PARALELNÉ OFARBOVANIE PERMUTÁCIÍ GRAFU

V tejto kapitole uvádzame metodiku a experimenty na potvrdenie alebo zamietnutie východiskovej hypotézy H1. Metodika a jej experimentálne overenie boli prezentované v rámci [III].

3.1.1 METODIKA ZAMERANÁ NA PARALELNÉ OFARBOVANIE PERMUTÁCIÍ GRAFU

Navrhujeme metodiku skúmania paralelného ofarbovania permutácií grafu:

- V prvom kroku potrebujeme nájsť vhodný **formát zápisu grafu** (resp. matice susedností grafu). Cieľom je nájsť taký formát zápisu grafu, ktorý zaberá najmenšie miesto v pamäti.
- Vstupný formát grafu (graf môže byť uložený napr. vo formáte graph6) **transformujeme** do tvaru, ktorý je potrebný pre funkcie programu (na matice susedností, v resp. dvojrozmerné polia naplnené celočíselnými hodnotami).
- V treťom kroku potrebujeme pre náš program zvoliť **vhodný model paralelného programovania.**
- V každej paralelnej časti programu (napr. vlákne) sa pre vstupný graf G vypočíta **náhodná permutácia matice** tohto grafu. Následne túto **permutáciu ofarbíme** pomocou algoritmu hranového backtrackingu. Pre každú ofarbovanú permutáciu matice susedností grafu G **odmeriame čas ofarbovania**. Výsledky zapíšeme do súboru zdieľaného medzi vláknami.

Posledný krok metodiky tvorí **test konzistentnosti ofarbovania** a jednoduchá **štatistika z behu programu**.

Aby sme pre naše potreby dostali dostatočne širokú vzorku nameraných výsledkov, budeme počítať s ofarbením 100 až 1000 permutácií vstupného grafu. Ide o časovo náročnú úlohu vhodnú pre **paralelné spracovanie**. Všetky vytvorené permutácie matice susedností grafu G môžu byť ofarbované súčasne, pričom sa časy ofarbovania všetkých permutácií zapíšu do zdieľaného súboru.

Hypotézu H1 overujeme pomocou vytvorenia aplikácie na paralelné ofarbovanie permutácií grafu. **Vstupom pre aplikáciu** je jeden kubický graf zadaný vo formáte **graph6 –** úsporný spôsob zápisu grafu založený na rozložení matice susedností grafu na niekoľko jednociferných a dvojciferných čísel.

Výstupom pre každú permutáciu zadaného grafu je:

- **čas ofarbovania** konkrétnej permutácie grafu meraný v milisekundách,
- hodnota **TRUE** ak je permutácia grafu regulárne ofarbiteľná troma farbami, **FALSE** v inom prípade,
- **kód permutácie** grafu odvodený z permutačnej matice susedností grafu – kód udáva na ktorom mieste v riadku permutačnej matice sa nachádza hodnota 1. Permutačná matica je naplnená nulami a obsahuje práve jednu 1 v každom riadku a stĺpci.

Samotný beh aplikácie sa skladá z troch hlavných krokov:

- Aplikácia pracuje s maticami susedností grafov, preto bolo ako prvé potrebné **preložiť vstup** z formátu graph6 na požadovaný formát. O toto prekladanie sa v aplikácii stará nami vytvorená funkcia, ktorej vstupom je počet vrcholov grafu a reťazec vo formáte graph6. Výstupom z tejto funkcie je matica susedností grafu.
- Matica susedností pôvodného grafu je ďalej rozposlaná na **paralelné spracovanie** pre určený počet **vlákien** aplikácie. Využívame model **POSIX threads** pričom ako parameter každému vláknu zadávame maticu susedností vstupného grafu. V každom z vlákien sa vykonáva niekoľko rôznych funkcií:
    1. **Vytvorenie permutácie matice** – túto funkciu sme zabezpečili implementáciou **konjugácie** podľa vzťahu 3.1, pričom permutačnú maticu P generujeme náhodne.
    2. **Ofarbenie permutácie matice –** na ofarbenie grafu bol použitý algoritmus **hranového backtrackingu**. Výstupom z tohto algoritmu je hodnota TRUE, ak je graf regulárne ofarbiteľný troma farbami alebo hodnota FALSE v inom prípade.
    3. **Zápis výsledku do zdieľaného súboru –** každé vlákno do zdieľaného súboru zapisuje čas, za ktorý prebehlo ofarbenie permutácie grafu, hodnotu ofarbovania TRUE alebo FALSE a kód permutácie matice susedností grafu.

- **Testy konzistentnosti ofarbovania –** po dokončení práce všetkých vlákien sa následne vykoná **test konzistentnosti ofarbovania** – test, ktorého výsledkom je hodnota KONZISTENTNÉ OFARBOVANIE ak je výsledok ofarbovania všetkých permutácií zadaného grafu homogénny – ak sú všetky permutácie grafu ofarbiteľné alebo všetky permutácie grafu neofarbiteľné, v inom prípade nadobúda hodnotu NEKONZISTENTNÉ OFARBOVANIE.

3.1.2 EXPERIMENTY ZAMERANÉ NA PARALELNÉ OFARBOVANIE PERMUTÁCIÍ GRAFU

Hypotézu sme overovali na grafe **Midzi 72**. Tento graf je snark, tak isto ako Petersenov graf. Skladá sa zo 72 vrcholov, pričom bol považovaný za graf s nadpriemerne dlhým časom ofarbovania, čo z neho urobilo vhodný cieľ pre naše testovanie [7].

V prvej inštancii sme spustili program na ofarbenie 100 permutácií grafu Midzi 72. V rámci Centra pre vysokovýkonné počítanie v Banskej Bystrici nám na tento výpočet postačovala lokálna, používateľská architektúra. V druhej inštancii sme spustili program s 500 permutáciami rovnakého problému a v tretej inštancii s 1000 permutáciami. Pre problémy s vyšším počtom permutácií matice susedností vstupného grafu sme potrebovali použiť rozšírenú architektúru. Konkrétne sa jednalo o použitie 15 výpočtových uzlov zo systému.

Pri každom behu programu sme merali najnižší čas ofarbovania grafu, najvyšší čas ofarbovania vstupného grafu a hodnotu mediánu pre ofarbovanie množiny permutovaných grafov.

Najkratšie časy ofarbovania grafov sa ukázali byť takmer zhodné – vo všetkých veľkostiach úlohy boli permutované grafy, ktorých čas ofarbovania bol najkratší < 1 milisekunda.Významné rozdiely v časoch ofarbovania sa ukázali až pri **porovnávaní maxím** (resp. mediánov) z časov výpočtov algoritmu.

Pri meraní sa ukázalo, že **doba výpočtu** algoritmu hranového backtrackingu je pre rôzne permutácie **výrazne odlišná**, čím sme vyvrátili pracovnú hypotézu H1. Namerané hodnoty sa pohybovali v intervale [0,3; 98] ms. Merania pre 100, 500 a 1000 permutácií sme zopakovali päťnásobne pre každú z veľkostí problému.

3.2 PARALELNÉ OFARBOVANIE PERMUTÁCIÍ MNOŽINY GRAFOV

Pri analýze výsledkov z meraní prezentovaných v kapitole 3.1.2 sme zistili, že exitujú také permutácie grafu Midzi 72, ktorých hranové ofarbovanie trvá menej ako milisekundu. Ak pre každý graf existuje také poradie ofarbovania hrán (resp. množina poradí) bolo by možné **minimalizovať čas ofarbovania** celých množín grafov práve zvolením takejto permutácie. Táto úvaha viedla k vytvoreniu pracovnej hypotézy H2.

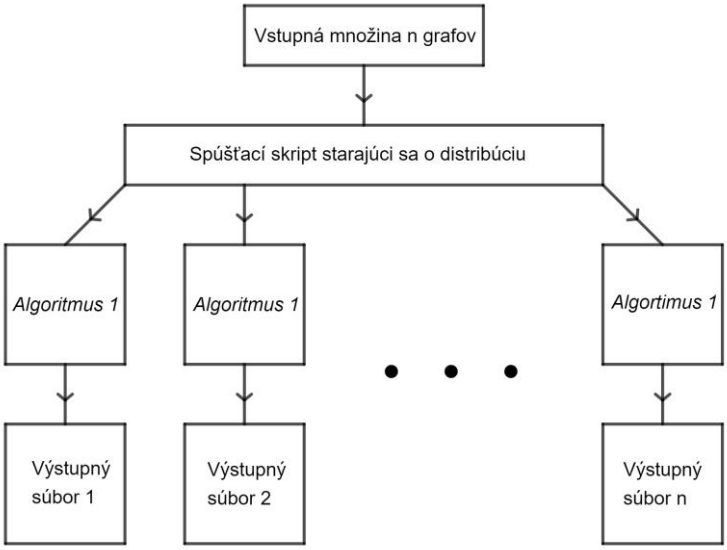
Pracovná hypotéza H2. *Neexistuje také poradie ofarbovania hrán grafu (permutácia), že po jeho aplikovaní na množinu grafov, bude čas ofarbenia celej množiny grafov redukovaný.*

Hypotéza H2 so sebou nesie predpoklad, že množina grafov, ktoré je potrebné hranovo ofarbiť bude **zhodnej veľkosti** – všetky grafy množiny majú rovnaký počet vrcholov a hrán.

Pre overenie pracovnej hypotézy H2 sme navrhli program, ktorý využíva kombináciu **distribuovaného a paralelného počítania**. Vstupom pre program je množina *n* grafov vo formáte graph6 – čo umožňuje použite **jedného vstupného súboru,** obsahujúceho množinu grafov. Z každého zo vstupných grafov paralelne ofarbujeme *p* permutácií. V jednom behu programu vypočítame ofarbiteľnosť pre *n*p* permutovaných grafov.

Program v toku distribuuje úlohy na jednotlivé stroje dostupného výpočtového systému, pričom každá z týchto úloh využíva paralelné vlákna na výpočet ofarbenia permutovaných grafov. Na obrázku 4.7 je náčrt schémy rozdeľovania úloh v programe.

Na paralelné ofarbovanie grafov v rôznom poradí hrán využívame algoritmus prezentovaný v podkapitole 3.1 – v schéme na obrázku 3.1 je samotný algoritmus označený ako **Algoritmus 1**.



Obrázok 3.1 – **Schéma rozdelenia úloh v distribuovanom programe pre ofarbovanie permutácií množiny grafov**

Výstupom z behu programu je *n* súborov - pre každý vstupný graf jeden súbor obsahujúci:

- **čas ofarbovania** permutácie grafu v milisekundách,
- hodnota **TRUE** ak je permutácia grafu regulárne ofarbiteľná troma farbami, **FALSE** v inom prípade,
- **kód permutácie** grafu odvodený z permutačnej matice susedností grafu a
- **počet rekurzívnych návratov** algoritmu hranového backtrackingu.

Keďže algoritmus hranového backtrackingu pri spätnom prefarbovaní hrán využíva rekurzívne funkcie, do výstupu z našej aplikácie sme pridali meranie počtu rekurzívnych návrtov ofarbovacieho algoritmu. Cieľom tohto merania je zistiť, či existuje spojitosť medzi časom hranového ofarbovania grafu a počtom rekurzívnych návratov ofarbovacieho algoritmu.

3.2.1 METODIKY ZAMERANÉ NA PARALELNÉ OFARBOVANIE PERMUTÁCIÍ MNOŽINY GRAFOV

Na algoritme, ktorý je predmetom tejto podkapitoly, sme navrhli metodiku a vykonali experimenty [VI, VII]. Cieľom týchto metodík je hľadanie **špecifickej permutácie** (resp. permutácie) grafu, ktorá môže byť použitá na **optimalizáciu času ofarbovania** vybranej množiny grafov. To znamená, že hľadáme poradie ofarbovania hrán grafu, ktoré po aplikovaní na celú množinu grafov zredukuje čas ofarbovania tejto množiny.

Navrhli sme tri experimenty v ktorých sme skúmali celkový čas výpočtu úlohy, minimálny a maximálny počet rekurzívnych volaní a minimálny a maximálny čas ofarbovania jednotlivých permutácií grafu. Ako vstup pre všetky experimenty sme použili množinu 34-vrcholových snarkov s cyklickou súvislosťou viac ako 5 z online databázy House of Graphs [7]. Táto množina grafov obsahuje 19 935 prvkov, pričom v jednotlivých experimentoch používame postupne narastajúce časti tohto datasetu.

Experimenty sú zamerané na potvrdenie alebo zamietnutie pracovnej hypotézy 2. Navrhnuté experimenty, ktoré pracujú s permutovanými grafmi s cieľom optimalizácie času ofarbovania vybranej množiny grafov sú:

- Ako vstup pre **experiment 1** používame dateset 9 967 500 grafov (19 935 pôvodných snarkov, pričom bol každý permutovaný 500 krát), ktoré boli výstupom z predošlého výskumu. Keďže množina výstupov bola komplexnejšia, ako sme v tomto experimente potrebovali, vybrali sme len údaj o čase ofarbenia pre všetkých 9 967 500 grafov. Z vybraných časov ofarbovania sme pripravili množinu dát s najnižším a množinu dát s najvyšším časom ofarbovania pre každý z pôvodných 19 935 grafov. Datasety sme vizualizovali a analyzovali. Výstupom experimentu 1 je nasledovné rozhodnutie: Generovanie 500 náhodných permutácií pri ofarbovaní množiny

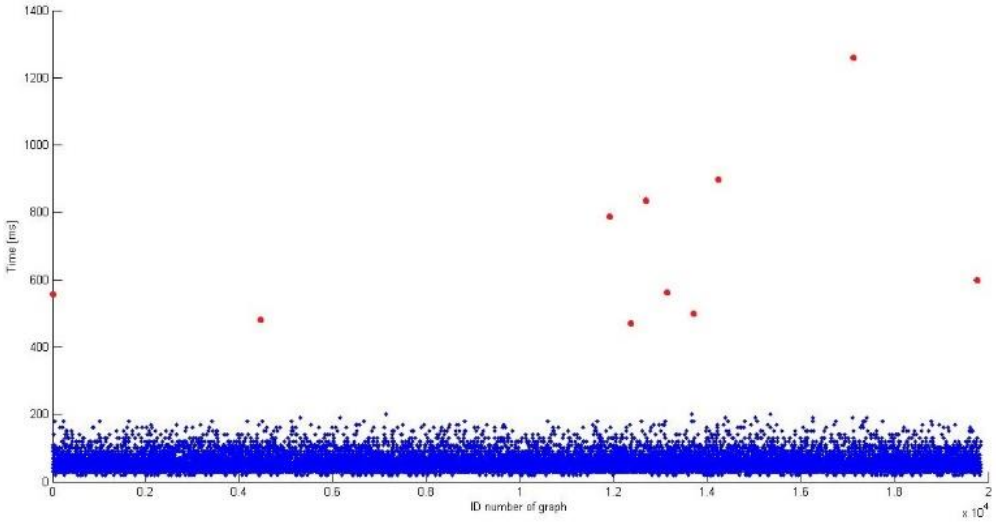
grafov vymeníme za pevne zvolených 500 permutácií s najnižším časom ofarbovania z predošlého merania.

- V **experimente 2** aplikujeme rozhodnutie experimentu 1 a zvolenú množinu permutácií s najnižším časom hranového ofarbovania aplikujeme na všetkých 19 935 grafov zo zvolenej množiny.
- Analogicky k experimentu 2, v **experimente 3** nahradíme generovanie 500 náhodných permutácií ofarbovaním grafov, ktoré permutujeme s použitím permutácie s najlepším časom ofarbovania. Túto permutáciu aplikujeme na vybranú množinu snarkov a meriame čas ofarbovania tejto množiny, ako aj jednotlivých grafov v nej.

EXPERIMENT 1

V experimente 1 uvádzame **najvyšší čas ofarbovania** zo všetkých 500 permutácií každého zo vstupných 19 935 grafov.

Z obrázku vidíme, že väčšina výpočtov ofarbovania trvala **v rozmedzí 20 až 200 milisekúnd**. Červenou farbou sme zvýraznili grafy, ktorých čas hranového ofarbovania bol extrémne dlhý.



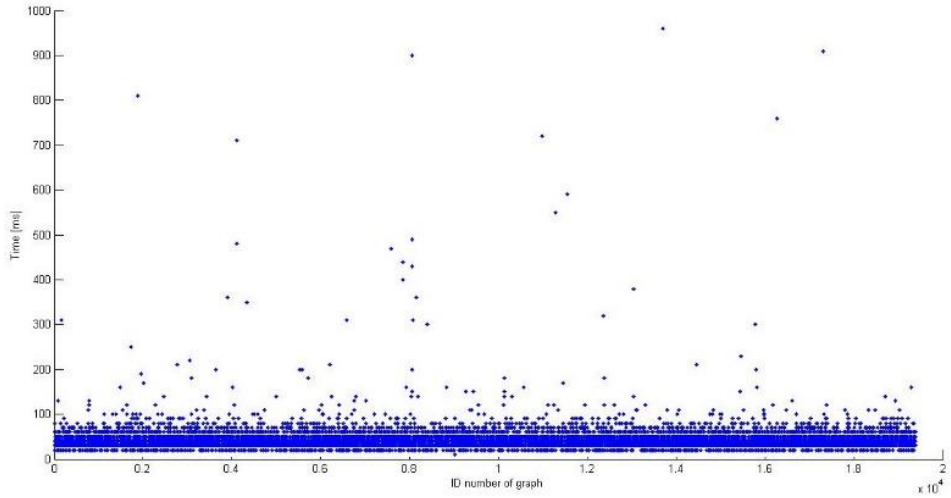
Obrázok 3.2 – Permutácie s maximálnym časom ofarbovania pre pôvodnú množinu 19 935 grafov

Tak isto ako sme vytvorili množinu dát s hodnotami pre najvyššie časy ofarbovania sme vytvorili dataset pre 19 935 **najkratších časov ofarbovania** – pre každý vstupný graf sme vybrali poradie hrán (permutáciu), ktoré bolo ofarbené v najkratšom čase. Všetky prvky tejto dátovej množiny sú permutácie s časom ofarbovania **kratším ako 1 milisekunda**.

EXPERIMENT 2

V experimente 2 sme vymenili generovanie náhodných 500 permutácií za **cielené permutovanie grafov** permutáciami, ktorých čas ofarbovania bol v predchádzajúcom experimente najnižší. Z predošlých meraní v experimente 1 sme vybrali množinu 500 permutácií, ktoré aplikujeme na pôvodnú množinu 19 935 grafov. Celú množinu grafov sme potom hranovo ofarbili.

Maximá časov ofarbovania pre každý graf z množiny sú vizualizované na obrázku 3.3.



Obrázok 3.3 – Maximálny čas ofarbovania pre vybraných 500 permutácií pôvodných 19 935 snarkov

Obrázok 3.3 poukazuje na dva zníženia v maximálnych časoch ofarbovania:

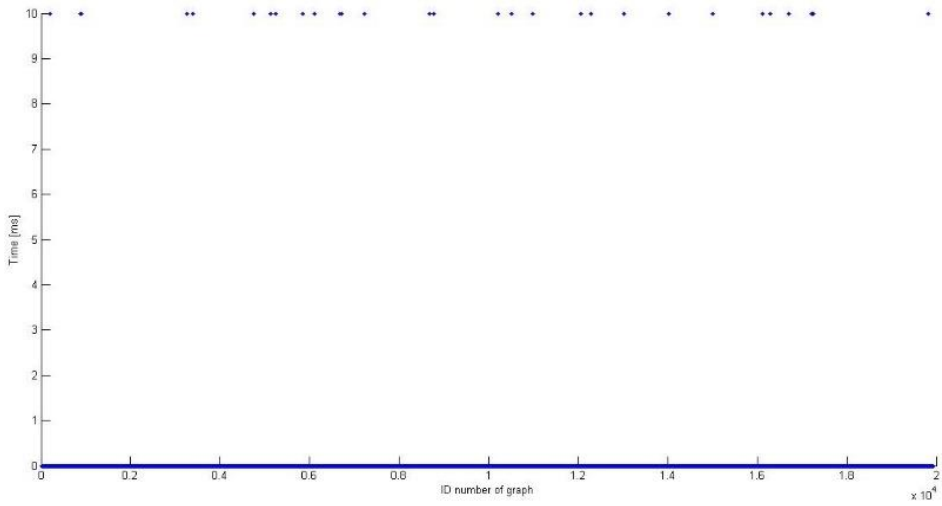
- Hlavná skupina grafov je hranovo ofarbená v čase medzi 20 a 100 milisekundami. V predošlom experimente bol tento časový interval medzi 20 a 200 milisekundami.
- Rozmedzie extrémne vysokých časov ofarbovania bolo znížené z 1260 milisekúnd na 960 milisekúnd.

EXPERIMENT 3

Z množiny 500 permutácií s najnižším časov ofarbovania sme zvolili permutáciu, ktorá má najnižší čas. Túto permutáciu sme aplikovali na pôvodnú množinu 19 935 grafov a ofarbili sme ich.

Obrázok 3.4, ktorý obsahuje výsledky z tohto experimentu, poukazuje na **pokles času hranového ofarbovania** na:

- hodnoty **pod 0,2 milisekundy** pre hlavnú skupinu grafov,
- a hodnoty v okolí **10 milisekúnd** v extrémnych prípadoch.



Obrázok 3.4 – Maximálny čas ofarbovania pre vybranú najlepšiu permutáciu pôvodných 19 935 snarkov

OPTIMALIZÁCIA ČASU HRANOVÉHO OFARBOVANIA MNOŽINY SNARKOV

V experimentoch 1 a 2 môžeme vidieť rôzne zníženia časov hranového ofarbovania. Algoritmus hranového backtrackingu je na vybranej množine snarkov **citlivý na počiatočnú hranu ofarbovania** (pracovná hypotéza H1).

V experimente 3 sme našli poradie hrán aplikovateľné na grafy z vybranej množiny, pomocou ktorého sme **znížili najvyššie časy hranového ofarbovania grafov** z pôvodného intervalu 20 až 1260 milisekúnd na 0,2 až 10 milisekúnd (tabuľka 3.1). Tento výsledok poukazuje na zamietnutie pracovanej hypotézy H2. Možnosť ďalšej časovej redukcie hranového ofarbovania grafov je skúmaná v podkapitole 3.3, kde tiež skúmame iné vlastnosti časovej minimalizácie, súvisiace s dekompozíciou problému.

Tabuľka 3.1 – Porovnanie najvyšších časov ofarbovania v experimentoch 6 - 8

	Experiment 1	Experiment 2	Experiment 3
Najvyššie časy ofarbovania pre hlavnú skupinu grafov [ms]	20 – 200	20 – 100	0,2 – 10
Maximálny čas ofarbovania [ms]	1260	960	10

3.3 PARALELNÉ OFARBOVANIE GRAFOV S VYUŽITÍM NAJLEPŠÍCH PERMUTÁCIÍ

Pracovná hypotéza H2 bola v predchádzajúcej časti práce zamietnutá, čo znamená, že vieme nájsť takú permutáciu grafu, ktorej aplikovaním na celú množinu grafov znížime čas ofarbovania celej množiny grafov. Ako bolo ukázané v experimente 8 pôvodná množina grafov bola po aplikovaní zvolenej permutácie rozdelená na dve skupiny grafov, ktoré môžeme špecifikovať ako:

- grafy ofarbiteľné pod 1 milisekundu,
- grafy ofarbiteľné za približne 10 milisekúnd.

Toto pozorovanie viedlo k vytvoreniu pracovnej hypotézy H3.

Pracovná hypotéza H3. *Nie je možné rozdeliť množinu grafov na podmnožiny (zhluky) tak, že pre každú podmnožinu grafov je možné nájsť také poradie ofarbovania hrán, ktoré znižuje čas potrebný na ofarbenie každého grafu v danej podmnožine.*

Pričom sa snažíme hľadať také ofarbenia grafov, ktoré **minimalizujú čas** samotného hranového ofarbovania. Pre naše potreby sme zvolili hranicu 1 milisekundy - požadujeme, aby algoritmus ofarbil každý graf zo zvolenej množiny v čase pod jednu milisekundu.

Táto podmienka je dôležitá nie len z hľadiska redukcie časovej náročnosti hranového ofarbovania množiny grafov, ale aj z pohľadu **dekompozície**. Ako bolo uvedené v podkapitole 1.3, zásadným konceptom pri dekompozícii úlohy na podúlohy je podobnosť (v ideálnom prípade rovnosť) času výpočtu všetkých podúloh. V pracovnej hypotéze H4 sa venujeme práve tejto podstatnej požiadavke na efektívnu dekompozíciu úlohy.

Pracovná hypotéza H4. *Neexistuje také poradie ofarbovania hrán grafu, že čas hranového ofarbovania je pre každý graf z množiny rovnaký (resp. je v stanovenej tolerancii - intervale).*

3.3.1 METODIKA PRE PARALELNÉ OFARBOVANIE GRAFOV S VYUŽITÍM NAJLEPŠÍCH PERMUTÁCIÍ

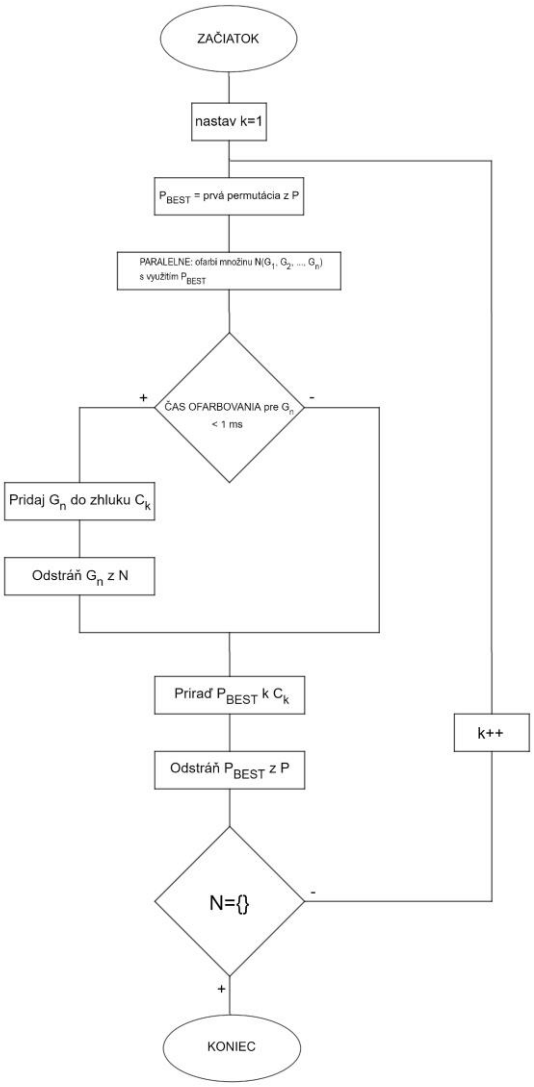
Pre potreby potvrdenia alebo zamietnutia pracovnej hypotézy H3 sme aplikovali niekoľko poznatkov. Keďže hypotéza hovorí o rozdeľovaní grafov do skupín, ako prvú sme na vybranú množinu grafov použili **konvenčnú zhlukovaciu metódu k-priemerov** (z anglického k-means). Tento algoritmus zhlukuje *n* prvkov do *k* zhlukov – v našom prípade ide o 19 935 prvkov (grafov) do 14 zhlukov, pričom počet zhlukov bol vybraný na základe takzvanej **metódy siluet** (z angl. Average silhouette method).

Pri analýze výsledkov tohto zhlukovania sme pozorovali, že grafy, ktorých hranové ofarbovanie trvalo viac ako 1 milisekundu boli umiestnené **na okrajoch zhlukov**. Tento fakt interpretujeme ako nešpecifickosť týchto grafov – môžeme povedať, že grafy, ktorých hranové ofarbovanie trvalo viac ako 1 milisekundu nie sú špecifické pre žiadny z vypočítaných zhlukov. Preto sme navrhli metodiku, ktorej vstupnými dátami sú dve množiny [IX]:

- **Množina permutácií *P***. Z už vypočítaných permutácií grafov vyberieme 500 takých, ktorých aplikácia na grafy viedla k najkratším časom ofarbovania.
- **Vstupná množina grafov *N*** zakódovaná vo formáte graph6.

Samotná metodika pracuje nasledovne:

- Z množiny *P* vyberieme **permutáciu s najnižším časom ofarbovania**, označíme ju ako P_{BEST} a aplikujeme ju na vstupnú množinu grafov *N*. Permutovanú množinu grafov následne paralelne hranovo ofarbíme rovnako ako v podkapitolách 3.1 a 3.2.
- Pre každý graf po ofarbení **overíme zvolenú podmienku**. Podmienku sme postavili na základne meraní z experimentu 8: **Bol graf G_n ofarbený v čase pod 1 milisekundu.**
    - **TRUE:** Graf G_n priradíme do zhluku C_k a odstránime ho z množiny *N*.
    - **FALSE:** Algoritmus pokračuje ďalším krokom bez akcie.
- Permutáciu P_{BEST} **priradíme ku zhluku** *Ck* a **odstránime** ju z množiny *P*.
- V prípade, že v množine *N* ešte ostávajú grafy, vytvoríme nový zhluk a postupujeme v algoritme od kroku 1. Ak je množina *N* prázdna, znamená to, že všetky grafy boli ofarbené v čase nižšom ako jedna milisekunda a algoritmus končí.



Obrázok 3.5 – Schéma algoritmu ofarbovania s využitím najlepších permutácií

Výstupom algoritmu je množina *k* súborov – každý súbor obsahuje zhluk ofarbených grafov s pridelenou permutáciou. Algoritmus je znázornený schémou na obrázku 3.5.

Využitím vytvorenej metodiky predpokladáme potvrdenie alebo zamietnutie **pracovných hypotéz H3 a H4,** definovaných v tejto podkapitole. Navrhnutý algoritmus **nedokončí svoj beh úspešne** v prípade, že nie je možné všetky grafy zo vstupnej množiny hranovo ofarbiť v čase nižšom ako jedna milisekunda. Pod úspešným dokončením behu programu v tomto prípade rozumieme vytvorenie *k* zhlukov, do ktorých sú rozdelené všetky grafy zo vstupnej množiny.

3.3.2 EXPERIMENTY ZAMERANÉ NA PARALELNÉ OFARBOVANIE GRAFOV A NAJLEPŠIE PERMUTÁCIE

Metodiku prezentovanú v podkapitole 3.3.1 sme experimentálne overili na rovnakej množine 19 935 snarkov ako pri predošlom algoritme. Z experimentu 8 predošlej metodiky sme vedeli, že táto množina grafov obsahuje veľkú časť, ktorá bude ofarbená v čase pod jednu milisekundu.

Vstupnú množina permutácií *P* sme vytvorili rovnako ako v experimente 7 predošlého algoritmu.

Obe vstupné množiny *N* a *P* sme zadali algoritmu na spracovanie. Z predošlého experimentu bolo zrejmé, že pri úspešnom behu programu bude výsledkom **2 až 40 zhlukov**. Prvý zhluk bude obsahovať všetky grafy, ktoré boli ofarbené v čase pod 1 milisekundu v experimente 8 – to znamená 19 896 zo vstupných 19 935 grafov. Zvyšných 39 grafov môže byť rozdelených do **1 až 39 zhlukov** podľa času ich hranového ofarbenia.

Algoritmus v experimente našiel nasledovné zhluky grafov s priradenými permutáciami:

- Zhluk 1, $P_{BEST} = P_1$
  - 19 896 grafov bolo hranovo ofarbených v čase nižšom ako 1 milisekunda,
  - N obsahuje 39 grafov neofarbených v čase nižšom ako 1 milisekunda.
- Zhluk 2, $P_{BEST} = P_2$
  - 39 grafov bolo hranovo ofarbených v čase nižšom ako 1 milisekunda,
  - N = { } a algoritmus skončil.

Na nájdenie zhlukov s ich priradenými permutáciami (postupnosťami ofarbovania hrán) boli potrebné len 2 cykly algoritmu. Výsledky z meraní spojených s experimentom a použitý výpočtový systém sú uvedené v tabuľke 3.2.

Tabuľka 3.2 – Výsledky meraní z experimentu zameraného na najlepšie permutácie

Výpočtový systém	5 uzlov, 12 procesorov na uzle, 4 GB RAM
Veľkosť problému	19 935 snarkov
Sekvenčný čas ofarbovania [hodiny]	3:22
Paralelný čas ofarbovania [hodiny]	Prvá skupina grafov: 0:31 Druhá skupina grafov: 0:02
Zrýchlenie výpočtu	6,122

Z experimentu vyplýva, že je možné nájsť množinu permutácií grafov, ktoré redukujú čas výpočtu celej množiny vstupných grafov na požadovanú úroveň. Zároveň je možné nájsť také poradie ofarbovania hrán grafu, že čas ofarbovania všetkých grafov z vybranej množiny je približne podobný – konkrétne pod jednu milisekundu. Tieto zistenia zamietajú hypotézy H3 a H4.

3.4 PARALELNÉ OFARBOVANIE GRAFOV S VYUŽITÍM NAJLEPŠÍCH PERMUTÁCIÍ A DÁVKOVÉHO SPRACOVANIA DÁT

Pre potreby spracovania množín grafov, ktoré obsahujú milióny grafov, sme vytvorili metodiku, ktorá spája princíp využitia vhodnej permutácie grafu a dávkového spracovania dát [IX].

3.4.1 METODIKA ZAMERANÉ NA PARALELNÉ OFARBOVANIE GRAFOV, NAJLEPŠIE PERMUTÁCIE A DÁVKOVÉ SPRACOVANIE DÁT

Metodika má na vstupe vybranú množinu grafov. Táto množina je predspracovacím skriptom rozdelená na maximálny možný počet častí obsahujúcich 500 000 grafov a jednu časť s ostatkom grafov. Každá takáto časť je skriptom distribuovaná na paralelné spracovanie v niekoľkých krokoch:

- **Načítanie dát z príslušnej časti do matice *M*.** Grafy sú uložené vo formáte graph6 čo umožňuje vytvoriť maticu, ktorá bude reprezentovať celú množinu vstupných dát. Počet riadkov tejto matice je rovný počtu grafov v danej časti, počet stĺpcov matice je daný veľkosťou grafu vo formáte graph6.
- **Paralelné ofarbovanie grafov.** Algoritmus spracováva grafy uložené v matici M v dávkach. Počet dávok je priamo závislí od počtu paralelných vlákien, ktoré sme schopní súčasne spustiť:

počet dávok = počet grafov v matici M / počet paralelných vlákien (3.2)

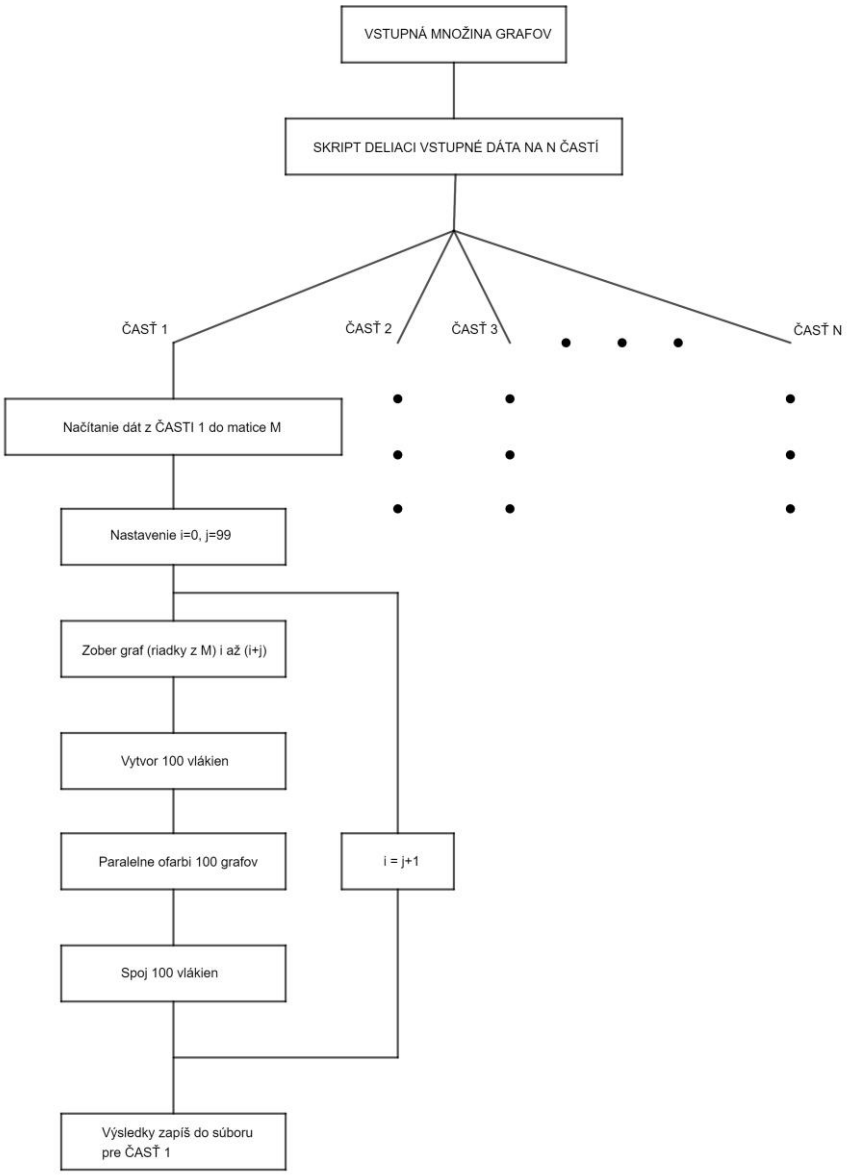
Algoritmus vytvorí daný počet paralelných vlákien, z ktorých každé dostane na vstupe jeden graf vo formáte graph6 a následne:

- graf preloží z formátu graph6 na maticu susedností,
- graf permutuje s využitím permutácie s najnižším časom ofarbovania nájdenej v 3.2,
- ofarbí graf.

Po ofarbení všetkých grafov v dávke sa vlákna spoja, algoritmus zoberie ďalšie grafy z matice *M* a opakuje tento krok.

- Po ofarbení všetkých grafov zo vstupnej časti dát algoritmus zapíše ID grafu a čas potrebný na hranové ofarbenie grafu do súboru, ktorý je spoločný pre všetky grafy v samotnej časti programu.

Výstupom z algoritmu je niekoľko súborov (počet súborov závisí od počtu častí, na ktoré je pôvodná množina dát rozdelená). Schéma algoritmu je uvedená na obrázku 3.6.



Obrázok 3.6 – Schéma paralelného programu zameraného na najlepšie permutácie a dávkové spracovanie dát

3.4.2 EXPERIMENTY ZAMERANÉ NA PARALELNÉ OFARBOVANIE GRAFOV, NAJLEPŠIE PERMUTÁCIE A DÁVKOVÉ SPRACOVANIE DÁT

Navrhnutý algoritmus sme experimentálne otestovali na množine grafových dát, ktorých časová náročnosť bola pri využití predchádzajúcich algoritmov príliš vysoká na to aby boli algoritmy použiteľné. Táto množina dát obsahuje **3 833 587 kubických grafov** s 34 vrcholmi. Množina bola rozdelená na sedem častí po 500 000 grafov a jednu časť s 333 587 grafmi. Samotné časti boli ofarbované v dávkach po 100 grafov. Výsledky merania a porovnanie časovej a pamäťovej náročnosti s algoritmom 2 prezentovaným v kapitole 3.2 sú uvedené v tabuľke 3.3.

Tabuľka 3.3 – Porovnanie časovej a pamäťovej náročnosti ofarbenia množiny 3 833 587 snarkov

	ALG2	**ALG4**	**ALG4**
Výpočtový systém	5/12ppn	5/12ppn	5/32ppn*
Čas ofarbovania [hh:mm:ss]	34:47:53	01:06:58	00:42:14
Potrebná pamäť RAM [kb]	3 450 900	994 598 552	993 818 164
Zrýchlenie výpočtu	-	31,178x	49,437x
Zvýšenie pamäťovej náročnosti	-	288,116x	287,988x

***virtualizované**

Z tabuľky 3.3 je zrejmé, že pri výpočtoch obsahujúcich milióny grafov dochádza k výraznému zrýchleniu výpočtu, ale zároveň sa dramaticky zvyšuje aj pamäťová náročnosť algoritmu. Pri využití rovnakého výpočtového systému sme dosiahli oproti algoritmu prezentovanému v podkapitole 3.2 približne **31,2 násobné zrýchlenie** výpočtu pričom sa pamäťová náročnosť zvýšila približne **288,1 násobne**.

3.5 PARALELNÉ OFARBOVANIE GRAFOV VYUŽÍVAJÚCE ZHLUKOVANIE GRAFOV A DÁVKOVÉ SPRACOVANIE DÁT

Na základe algoritmov prezentovaných v kapitolách 4.3 a 4.4 sme navrhli **kombináciu** týchto algoritmov [IX], ktorá zabezpečuje zníženie času výpočtu ofarbovania množiny grafov a splní podmienku, ktorá vyplýva z hypotézy 4 (rovnakosť resp. podobnosť času ofarbovania všetkých grafov z vybranej množiny).

3.5.1 METODIKA ZAMERANÁ NA PARALELNÉ OFARBOVANIE GRAFOV, ZHLUKOVANIE GRAFOV A DÁVKOVÉ SPRACOVANIE DÁT

Ako **vstup pre algoritmus** sme zvolili rovnaké množiny ako pri algoritme prezentovanom v 3.3 – množinu grafov N a množinu permutácií P. Algoritmus paralelného ofarbovania grafov využívajúci zhlukovanie grafov a dávkové spracovanie dát je opísaný nasledovne:

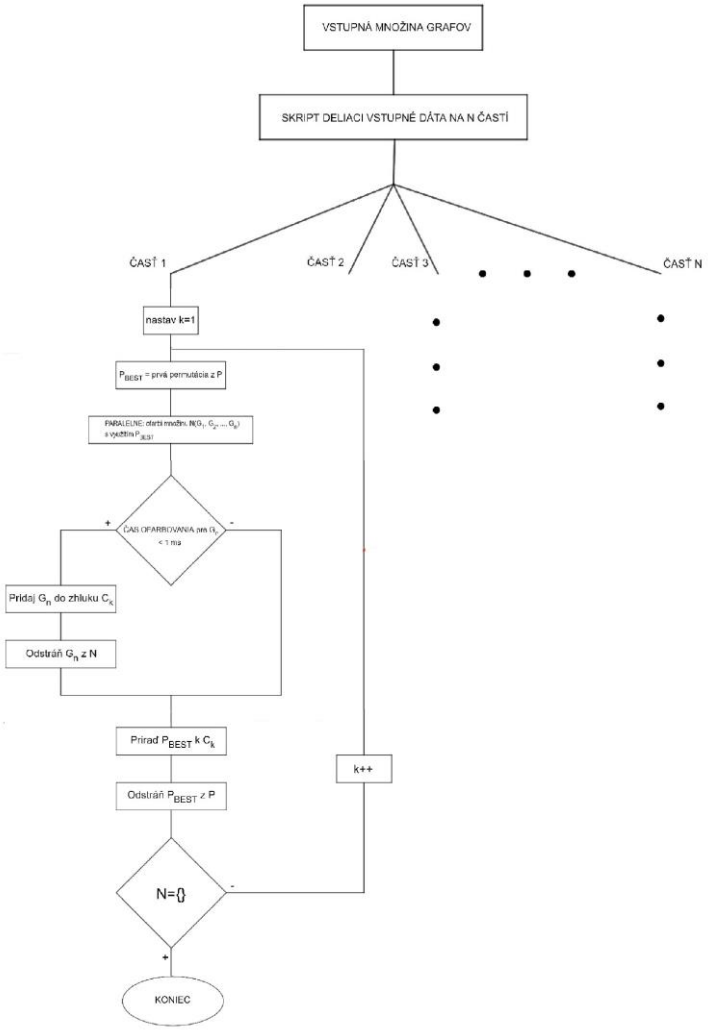
- **Rozdelenie množiny grafov *N* na dátové časti.** Rovnako, ako v prípade algoritmu zameraného na dávkové spracovanie dát (kapitola 3.4), algoritmus pomocou rozdeľovacieho skriptu rozdelí vstupnú množinu grafov N na n menších častí. Presnejšie, skript rozdelí vstupné dáta na približne rovnako veľké podmnožiny obsahujúce cca. 500 000 prvkov (grafov).
- **Distribuovanie dátových častí.** Po rozdelení množiny grafov na dátové časti postupujeme podľa algoritmu prezentovaného v podkapitole 3.4. Jednotlivé dátové časti (podmnožiny) sú distribuované na dostupný, špecifikovaný výpočtový systém.
- V ďalšom kroku navrhnutý algoritmus vykoná jednotlivé kroky algoritmu pre paralelné ofarbovanie grafov s využitím najlepších permutácií (prezentovaného v podkapitole 3.3). Pre každú z n dátových častí vytvorených zo vstupnej množiny grafov N:
    - **Výber permutácie s najnižším časom ofarbovania** P_{BEST} zo vstupnej množiny permutácií P a jej následné aplikovanie na celú dátovú časť podľa vzťahu 3.1.

- **Paralelné hranové trojofarbenie grafov** v dátovej časti s využitím výpočtového modelu paralelných vlákien.
- **Overenie podmienky** pre všetky grafy v dátovej časti. Podmienka je zostavená nasledovne: Bol graf *G* hranovo ofarbený v čase nižšom ako jedna milisekunda?
- **Vytvorenie zhluku grafov**, ktoré splnili podmienku uvedené v predchádzajúcom kroku.
- **Pridelenie P_{BEST}** k vytvorenému zhluku grafov.
- **Odstránenie P_{BEST}** z množiny permutácií P, odstránenie grafov s časom ofarbovania nižším ako jedna milisekunda z dátovej časti.

Tieto kroky sú opakované kým daná dátová časť nie je prázdna alebo pokiaľ algoritmus neúspešne nevyskúša všetky permutácie zo vstupnej množiny *P* (takáto situácia môže nastať v prípade, že sa v množine *N* nachádza graf, ktorý nemôže byť ofarbený v čase nižšom ako jedna milisekunda).

Výstup z výpočtu algoritmu sa skladá z množiny súborov, ktorá obsahuje zhluky grafov s ich pridelenými permutáciami pre každú z vytvorených *n* dátových častí. Na získanie celkových výsledkov z výpočtu algoritmu je potrebné na množinu súborov použiť aglomeračnú funkciu. Táto funkcia spája súbory (zhluky), ktorým bola v behu programu pridelená rovnaká permutácia.

Schéma algoritmu, ktorý využíva paralelné ofarbovanie grafov, zhlukovanie grafov a dávkové spracovanie dát je prezentovaná na obrázku 3.7.



Obrázok 3.7 - Schéma paralelného programu zameraného na najlepšie permutácie, dávkové spracovanie dát a zhlukovanie grafov

3.5.2 EXPERIMENTY PRE PARALELNÉ OFARBOVANIE GRAFOV, ZHLUKOVANIE GRAFOV A DÁVKOVÉ SPRACOVANIE DÁT

Cieľom experimentov uvedených v tejto podkapitole je overenie funkčnosti algoritmu navrhovaného v 3.5.1 pričom sa sústredíme na minimalizáciu počtu vytvorených zhlukov. Na úvod sme funkčnosť a vlastnosti algoritmu navrhovaného v 3.5.1 experimentálne overili na dvoch rozmerných množinách dát:

- množina 3 833 587 grafov, ktorá obsahuje 34 vrcholové snarky,
- množina 25 286 953 snarkov s 34 vrcholmi.

Namerané výsledky z týchto experimentov sú uvedené v tabuľkách 3.4 a 3.5. Všetky uvádzané hodnoty boli namerané na výpočtovom systéme pozostávajúcom z 6 uzlov, pričom bolo na každom uzle 12 procesorov. Z predchádzajúcich experimentov (najmä toho, uvedeného v 3.4.1), bolo zrejmé, že algoritmus založený na dávkovom spracovaní dát má vysoké požiadavky na pamäť systému. V prípade 3 833 587 snarkov algoritmus potreboval takmer 1 TB pamäte, v prípade 25 286 953 grafov algoritmus zužitkoval 4 TB pamäte.

V nižšie uvedených tabuľkách uvádzame merania pre nasledovné vlastnosti výpočtov:

- **Číslo zhluku** - ID číslo zhluku, ktorý obsahuje dané grafové dáta,
- **Číslo P_{BEST}** - ID číslo permutácie priradenej k danému zhluku,
- **Čas výpočtu** – čas ofarbovania a režijné náklady pre všetky grafy v danom zhluku,
- **Potrebná pamäť** – pamäť potrebná pri ofarbovaní daného zhluku,
- **Počet grafov ofarbených** v čase vyššom ako jedna milisekunda. Tieto grafy je potrebné ofarbiť s využitím inej permutácie.

Tabuľka 3.4 – Zhluky grafov množiny 3 833 587 snarkov

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	01:06:58	994.598	532 364
2	2	00:07:00	32.059	57 274
3	3	00:00:49	1.457	5 775
4	4	00:00:07	0.254	0

Tabuľka 3.5 – Zhluky grafov množiny 25 286 953 snarkov

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	04:19:09	3219.341	1 356 175
2	2	00:11:43	12.242	127 775
3	3	00:01:26	3.362	15 032
4	4	00:00:16	0.845	6
5	5	00:00:01	0.017	0

3.5.3 EXPERIMENTY PRE PARALELNÉ OFARBOVANIE GRAFOV, ZHLUKOVANIE GRAFOV A DÁVKOVÉ SPRACOVANIE DÁT NA SADÁCH SNARKOV S VÄČŠÍM POČTOM VRCHOLOV

Pre ďalšie overenie algoritmu a princípu ofarbovania permutovaných grafov (a zároveň overenia hypotéz H1 - H4 na iných množinách snarkov) sme pracovali aj s množinami 36, 38 a 40 vrcholových snarkov. Pre každú z týchto množín sme postupovali nasledovne:

- **Náhodné permutovanie množín grafov** – pre grafy s 36, 38 a 40 vrcholmi potrebujeme nájsť permutácie, ktorých aplikovanie na množinu grafov znižuje čas ofarbenia grafov v množine. Postupovali sme rovnako v podkapitole 3.2.
- **Vytvorenie množiny najlepších permutácií** – podobne ako v podkapitole 4.2 sme pre množinu 36, 38 a 40 vrcholových grafov náhodným permutovaním našli množinu 500 najlepších permutácií, ktorých aplikovanie na množinu grafov znižovalo čas ofarbenia všetkých grafov v množine.

- **Využitie dávkového spracovania grafov a zhlukovania** – po nájdení vhodného počtu permutácií sme využili algoritmus prezentovaný v 3.5.1 a ofarbili sme všetky podmnožiny 36, 38 a 40 vrcholových snarkov.

V tabuľke 3.6 uvádzame počet grafov v jednotlivých podmnožinách 36, 38 a 40 vrcholových snarkov. Snarky jednotlivých veľkostí sú do týchto podmnožín rozdelené na základe matematických vlastností jednotlivých grafov, tzv. veľkosti najmenšej kružnice v grafe a cyklickej súvislosti. Pri množinách označených znakom '+' zatiaľ nebolo možné vygenerovanie všetkých grafov danej veľkosti (resp. daných vlastností).

Tabuľka 3.6 – Podmnožiny 36, 38 a 40 vrcholových snarkov

Počet vrcholov grafu	Veľkosť najmenšej kružnice ≥ 4	Veľkosť najmenšej kružnice ≥ 5	Veľkosť najmenšej kružnice ≥ 6	Cyklická súvislosť ≥ 5
36	404 899 916	60 167 732	1	180 612
38	-	19 775 763+	39	35 429+
40	-	-	25	-

Začali sme podmnožinou 36 vrcholových grafov s cyklickou súvislosťou vyššou alebo rovnou 5. Táto množina obsahuje 180 612 grafov a pomocou nášho algoritmu sme ju rozdelili do štyroch zhlukov (tabuľka 3.7).

Tabuľka 3.7 – Zhluky grafov množiny 180 612 snarkov s 36 vrcholmi

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	00:01:10	1.326	4008
2	2	00:00:07	0.839	241
3	3	00:00:07	0.376	12
4	4	00:00:01	0.017	0

Pri 36 vrcholových snarkoch existuje množina grafov s veľkosťou najmenšej kružnice v grafe rovnou alebo vyššou ako 6, ktorá obsahuje jeden graf. Tento graf bol ofarbený v čase nižšom ako jedna milisekunda pomocou permutácie $P_{BEST} = 2$. Pokračovali sme množinou 60 167 732 snarkov s 36 vrcholmi. Túto množinu náš algoritmus rozdelil na 7 zhlukov (tabuľka 3.8).

Tabuľka 3.8 – Zhluky grafov množiny 60 167 732 snarkov s 36 vrcholmi

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	08:21:18	4372.881	4 229 360
2	2	00:34:31	35.237	256 902
3	3	00:03:11	1.245	53 487
4	4	00:01:07	0.891	16 343
5	5	00:00:33	0.873	8 901
6	6	00:00:31	0.754	8 042
7	7	00:00:04	0.019	0

Najrozsiahlejšia množina 36 vrcholových snarkov obsahuje 404 899 916 prvkov. Túto množinu náš algoritmus spravoval do 16 zhlukov (tabuľka 3.9).

Tabuľka 3.9 – Zhluky grafov množiny 404 899 916 snarkov s 36 vrcholmi

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	47:16:14	8472.230	17 642 937
2	2	03:32:49	56.671	12 876 022
3	3	00:56:11	30.290	4 038 765

4	4	00:40:26	12.128	1 844 397
5	5	00:38:39	9.133	1 000 036
6	6	00:07:12	2.465	321 009
7	7	00:05:31	1.026	162 932
8	8	00:00:22	0.891	21 329
9	9	00:00:20	0.806	17 890
10	10	00:00:18	0.650	12 993
11	11	00:00:17	0.521	9 636
12	12	00:00:13	0.360	4109
13	13	00:00:10	0.332	1572
14	14	00:00:03	0.196	60
15	15	00:00:02	0.183	38
16	16	00:00:03	0.067	0

Pokračovali sme množinami snarkov, ktoré sa skladajú z 38 vrcholov. K dispozícií sú 3 takéto množiny – množina s 35 429 grafmi, množina s 39 grafmi a množina s 19 775 768 prvkami. Výsledky rozdelenia do zhlukov pre tieto množiny grafov sú uvedené v tabuľkách 3.10, 3.11 a 3.12.

Tabuľka 3.10 – Zhluky grafov množiny 35 429 snarkov s 38 vrcholmi

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	00:00:42	0.630	1053
2	2	00:00:11	0.439	0

Tabuľka 3.11 – Zhluky grafov množiny 39 snarkov s 38 vrcholmi

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	00:00:03	0.371	0

Tabuľka 3.12 – Zhluky grafov množiny 19 775 768 snarkov s 38 vrcholmi

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	03:16:02	3018.03	594 945
2	2	00:01:33	1.139	10 709
3	3	00:00:03	0.732	4 283
4	4	00:00:01	0.424	0

Poslednou množinou snarkov sú 40 vrcholové grafy, ktorých je 25. Aj napriek nízkemu počtu týchto grafov, bola táto množina rozdelená do troch zhlukov uvedených v tabuľka 3.13.

Tabuľka 3.13 – Zhluky grafov množiny 25 snarkov so 40 vrcholmi

Číslo zhluku	Číslo P_{BEST}	Čas výpočtu [hh:mm:ss]	Potrebná pamäť [GB]	Počet grafov ofarbených > 1 ms
1	1	00:00:04	0.230	9
2	2	00:00:01	0.089	2
3	3	00:00:01	0.019	0

3.5.4 VYHODNOTENIE EXPERIMENTOV PRE PARALELNÉ OFARBOVANIE GRAFOV, ZHLUKOVANIE GRAFOV A DÁVKOVÉ SPRACOVANIE DÁT

Celkový čas výpočtu potrebný na ofarbenie celej množiny grafov pričom prihliadame na podmienku plynúcu z hypotéz H3 a H4 môžeme vypočítať nasledovne:

$$T_o = \sum_{i=1}^{k} Tc_i \qquad (3.3)$$

kde *To* je celkový čas výpočtu ofarbovania danej množiny grafov, *k* je počet zhlukov a Tc_i je výpočtový čas pre zhluk číslo *i*. Takto vypočítaný celkový čas ofarbovania uvádzame v tabuľke 3.14.

Aj keď algoritmus pre svoj beh vyžaduje vysokú pamäťovú náročnosť, umožňuje paralelné zrýchlenie výpočtu a vytvára zhluky, ktoré majú pridelené permutácie, pomocou ktorých vieme graf ofarbiť uspokojivo z pohľadu hypotézy H4. Ide teda o **efektívnu dekompozíciu problému**.

V tabuľkách 3.7 – 3.13 môžeme vidieť, že nie je potrebné vysoké množstvo vhodne zvolených permutácií na to aby sme dosiahli čas ofarbovania každého grafu z ofarbovanej množiny nižší ako jedna milisekunda. Konkrétne počty permutácií (resp. klastrov) sú uvedené v tabuľke 3.14.

Tabuľka 3.14 – Počet potrebných zhlukov a celkový čas ofarbovania pre všetky testované množiny snarkov

34 vrcholové snarky				
Počet grafov	19 935	-	3 833 587	25 286 953
Počet zhlukov	2	-	4	5
Celkový čas ofarbovania	00:00:33	-	01:14:54	04:32:35
36 vrcholové snarky				
Počet grafov	180 612	1	60 167 732	404 899 916
Počet zhlukov	4	1	7	16
Celkový čas ofarbovania	00:01:25	00:00:01	09:01:15	53:18:50
38 vrcholové snarky				
Počet grafov	35 429	39	19 775 763+	-
Počet zhlukov	2	1	4	-
Celkový čas ofarbovania	00:00:53	00:00:03	03:17:39	-
40 vrcholové snarky				
Počet grafov	-	25	-	-
Počet zhlukov	-	3	-	-
Celkový čas ofarbovania	-	00:00:06	-	-

ZÁVER

Hranové ofarbovanie grafov je operácia, ktorá je často opakovanou časťou výpočtu pri skúmaní rôznych vlastností grafov, ako napríklad hľadanie hranovej a vrcholovej kritickosti alebo kokritickosti grafu. Hlavným cieľom práce bol návrh, implementácia a overenie algoritmov, metodík a nástrojov, ktoré prispejú k optimalizácii dekompozície paralelných aplikácií pri práci s hranovým ofarbovaním kubických grafov. Konkrétnymi cieľmi práce bolo potvrdenie alebo zamietnutie nasledujúcich hypotéz:

- *Algoritmus hranového backtrackingu nie je citlivý na počiatočný vrchol ofarbovania grafu. To znamená, že ak pri výpočte použijeme rôznu postupnosť hrán bude doba výpočtu algoritmu pre zadaný graf rovnaká.*
    - Vo všetkých experimentoch uvedených v kapitole 3 sme ukázali, že doba výpočtu algoritmu hranového backtrackingu je pre rôzne permutácie **výrazne odlišná**, čím sme hypotézu zamietli. Namerané hodnoty sa pohybovali v intervale od časov pod 1 milisekundu až po časy približne 1300 ms.
- *Neexistuje také poradie ofarbovania hrán grafu (permutácia), že po jeho aplikovaní na množinu grafov, bude čas ofarbenia celej množiny grafov redukovaný.*
    - Našli sme permutáciu aplikovateľnú na grafy z vybranej množiny, pomocou ktorej sme **znížili časy hranového ofarbovania grafov.** Tento výsledok bol experimentálne overený v podkapitole 3.2.2 a poukazuje na zamietnutie pracovanej hypotézy H2.
- *Nie je možné rozdeliť množinu grafov na podmnožiny tak, že pre každú podmnožinu grafov sme schopní nájsť jej vlastné poradie ofarbovania hrán, ktoré znižuje čas potrebný na ofarbenie každého grafu v danej podmnožine.*

- Navrhli, implementovali a experimentálne sme overili algoritmus, pomocou ktorého sme našli množinu permutácií grafov, ktoré **redukujú čas výpočtu** celej množiny vstupných grafov na požadovanú úroveň. Experimenty zhlukovania grafov sú uvedené v podkapitole 3.3.2.

- *Neexistuje také poradie ofarbovania hrán grafu, že čas hranového ofarbovania je pre každý graf z množiny rovnaký.*
  - Je možné nájsť také poradie ofarbovania hrán grafu, že čas ofarbovania všetkých grafov z vybranej množiny je približne podobný – v našom prípade sme v experimentoch uvedených v kapitole 3.3.2 **použili množinu permutácií**, pomocou ktorých sme ofarbili jednotlivé grafy z vybranej množiny **pod jednu milisekundu.** Čím sme optimalizovali dekompozíciu paralelných a distribuovaných výpočtov čo bol hlavný cieľ tejto práce.

Okrem cieľov práce formalizovaných v hypotézach sme preskúmali možnosti **paralelizácie výpočtov** týkajúcich sa hranového ofarbovania grafov. V rámci práce sme optimalizovali (minimalizovali) čas ofarbovania množín vybraných grafov s využitím vhodne zvolených permutácií grafov. **Dekomponovali** sme problém ofarbenia rozmernej množiny grafov na menšie podproblémy, ktoré boli distribuované na výpočtovú architektúru a počítané s využitím paralelných algoritmov. Pomocou algoritmu prezentovaného v podkapitole 3.5 sme vypočítali hranové trojofarbenie všetkých veľkých množín snarkov z portálu House of Graphs tak, aby sme zabezpečili podobný (resp. rovnaký) čas ofarbovania pre každý graf z vybranej množiny – čas nižší ako jedna milisekunda. Táto podobnosť (resp. rovnosť) času je kľúčovým konceptom efektívnej dekompozície akéhokoľvek problému na menšie podproblémy. Výpočty tak môžu byť vykonávané na akomkoľvek výpočtovom systéme a trvanie ofarbenia jednotlivých grafov bude vždy podobné (resp. rovnaké). Tým sme ukázali, že využitie paralelných a distribuovaných výpočtov je možné do istej **úrovne granularity danej úlohy**.

Využitie paralelného výpočtového systému v kombinácii s paralelným programovaním pre potreby problému ofarbovania množín grafov je vhodné a dôležité. Na výpočtovom systéme, ktorý obsahoval len fyzické zdroje sme výpočet tohto problému **niekoľkonásobne zrýchlili**.

Na záver výskumnej časti práce sme uviedli koncept **zhlukovania grafov** na základe permutácií. Štandardné metódy zhlukovania potrebujú pre svoj beh poznať počet zhlukov, do ktorých majú byť dáta rozdelené. V prípade grafových dát, s ktorými sme v práci pracovali je počet zhlukov ťažko odhadnuteľný. Využili sme preto interpolačné funkcie na odhadnutie vhodného počtu potrebných zhlukov a tým sme otvorili nové potenciálne oblasti skúmania vo zvolenej problematike.

ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV

[1] KOLLÁR, J. 2003. *Metódy a prostriedky pre výkonné paralelné výpočty*. Košice : Technická Univerzita v Košiciach, 2003. 100 s. ISBN 80-89066-70-4

[2] COULOURIS, G., DOLLIMORE, J., KINDBERG, T., BLAIR, G. *Distributed systems – concepts and designs*. 2012. Addison-Wesley, 5. Vydanie, 1065 strán. ISBN 978-0-13-214301-1

[3] GRAMA, A., GUPTA, A., KARYPIS, G., KUMAR, V. *Introduction to Parallel Computing*. 2003. Addison Wesley, Druhé vydanie, 856 strán. ISBN: 978-0201648652

[4] DIESTEL R. 2016. *Graph theory*. Piate vydanie. Springer - Verlag, Heidelberg, 2016, 447 s. ISBN 978-3-662-53621-6

[5] BRINKMANN, G., GOEDGEBEUR J., HÄGGLUND J., MARKSRÖM K.: Generation and properties of snarks. 2013. Journal of Combinatorial Theory, Series B, Volume 103, Issue 4, s. 468-488, ISSN 0095-8956

[6] KARABÁŠ, J., MÁČAJOVÁ, E., NEDELA, R.: 6-decomposition of snarks. 2013. In European Journal of Combinatorics: 20th International workshop on combinatorial algorithms (IWOCA), Hradec nad Moravicí, 28 June - 2 July, 2009. Elsevier, 2013, Volume 34, Issue 1, s. 111-122. ISSN 0195-6698

[7] BRINKMANN, G., COOLSAET, K., GOEDGEBEUR, J., MÉLOT H.: House of Graphs: a database of interesting graphs, Discrete Applied Mathematics, 161:311-314, 2013. Available at http://hog.grinvin.org

ZOZNAM AUTORSKÝCH PUBLIKÁCIÍ

[I.] ŠKRINÁROVÁ, J., **DUDÁŠ, A.**, VESEL, E.: Model of education and training strategy for the management of HPC systems. In 2017 IEEE 14th International Scientific Conference on Informatics, Poprad, Slovakia, 14. – 17. November 2017. ISBN 978-1-5386-0889-0

[II.] ŠKRINÁROVÁ, J., **DUDÁŠ, A.**: A Methodology for the professional training of the management and evaluation of HPC systems. Open Computer Science, Volume 8, Issue 1, s. 68-79, ISSN 2299-1093

[III.] **DUDÁŠ, A.**, ŠKRINÁROVÁ, J., VOŠTINÁR, P., SILÁČI, J.: Improved process of running tasks in the high performance computing. In ICETA 2018: Proceedings: 16th IEEE International Conference on Emerging eLearning Technologies and Applications. - New Jersey: Institute of Electrical and Electronics Engineers. - ISBN 978-1-5386-7912-8. - s. 133-140

[IV.] ŠKRINÁROVÁ, J., **DUDÁŠ, A.**: Procesy operačného systému vo vyučovaní informatiky. In Didinfo 2019: medzinárodná konferencia o vyučovaní informatiky: 25. ročník konferencie. - ISSN 2454-051X (online). - 1. vyd. - Banská Bystrica: Univerzita Mateja Bela v Banskej Bystrici, 2019. - ISBN 978-80-557-1533-9 (online). - s. 152-156

[V.] CHOVANCOVÁ, O., PIATRIKOVÁ, L., **DUDÁŠ, A.**: Improving fuzzy c-means algorithm using particle swarm optimalization and fuzzy c-means++.
In Information and digital technologies 2019: proceedings of the international conference. - 1. vyd. - Danvers: Institute of Electrical and Electronics Engineers, 2019. - ISBN 978-1-7281-1401-9. - s. 173-179.

[VI.] **DUDÁŠ, A.**, ŠKRINÁROVÁ, J., VESEL, E.: Optimization design for parallel coloring of a set of graphs in the High-Performance Computing. In 2019 IEEE 15th International Scientific Conference on Informatics, Poprad, Slovakia, 20. – 22. November 2019.

[VII.] VESEL, E., ŠKRINÁROVÁ, J., **DUDÁŠ, A.**: Comparison of in-house HPC calculation with public cloud computing for parallel algorithm containing recursive functions. In ICETA 2019 : Proceedings : 17th IEEE International Conference on Emerging eLearning Technologies and Applications. - New Jersey: Institute of Electrical and Electronics Engineers.

[VIII.] CHOVANCOVÁ, O., LAUKOVÁ, J., **DUDÁŠ, A.**, KOSTOLNÝ, J.: Interactive Data Visualization of Fuzzy Membership Functions and Fuzzy Clustering. In ICETA 2019 : Proceedings : 17th IEEE International Conference on Emerging eLearning Technologies and Applications. - New Jersey: Institute of Electrical and Electronics Engineers.

V recenznom konaní:

[IX.] ŠKRINÁROVÁ, J., **DUDÁŠ, A.**: Optimization of Functional Decomposition of Parallel and Distributed Computations in Graph Coloring with the use of High Performance Computing. In Computing and Informatics (CAI), 2020. ISSN 1335-9150 *(podané: 29.2.2020)*

[X.] **DUDÁŠ, A.,** ŠKRINÁROVÁ, J.: Edge coloring of set of graphs with the use of data decomposition and clustering. In IPSI BgD Transactions on Internet Research. ISSN 1820 - 4503 *(podané: 5.5.2020)*