VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Paralelní metody hraničních prvků

Parallel Boundary Element Methods

2013

Michal Kravčenko

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání diplomové práce

Student: **Bc. Michal Kravčenko**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 1103T031 Výpočetní matematika

Téma: Paralelní metody hraničních prvků
Parallel Boundary Element Methods

Zásady pro vypracování:

Metody hraničních prvků je efektivním nástrojem pro řešení okrajových úloh parciálních diferenciálních rovnic. Její efektivita spočívá v redukci neznámých na hranici a je zvláště vhodná pro vnější úlohy. Nevýhodou metody je, že výsledné matice soustav lineárních rovnic jsou husté. Naštěstí byly nedávno vyvinuty zřeďovací techniky. Jejich paralelní implementace je však stále nejasná, neboť není snadné efektivně distribuovat zároveň bloky matice a s ní související diskretizaci hranice. Ukazuje se, že k tomu lze využít nedávné výsledky teorie grafů.
Student má za úkol:
- studovat principy metody hraničních prvků,
- implementovat v prostředí MPI distribuované matice a vektory,
- implementovat distribuci matice pro dodané faktorizace grafů,
- provést studie škálovatelnosti na paralelním clusteru.

Seznam doporučené odborné literatury:

O. Steinbach, S. Rjasanow - The Fast Solution of Boundary Boundary Integral Equations. Springer Verlag, 2007.
M. Bebendorf - Hierarchical Matrices. Springer Verlag, 2008.

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

Vedoucí diplomové práce: **doc. Ing. Dalibor Lukáš, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

doc. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 .....

Především bych chtěl poděkovat svému vedoucímu Doc. Ing. Daliboru Lukášovi, Ph.D. za velmi ochotné a časté konzultace, díky kterým byla tato práce dokončena v termínu. Rovněž děkuji své mamince za to, že mi umožnila zpracování této práce tolerováním mého věčného studia.

Abstrakt

Předmětem této práce je studium a minimalizace paměťových nároků paralelního řešení soustav lineárních rovnic reprezentující kolokačně diskretizované hraniční integrální rovnice získané metodou hraničních prvků. Je představeno řešení soustav diferenciálních rovnic Metodou Hraničních Prvků využívající kolokační disktretizační metodu. Paralelizace řešení získáné soustavy rovnic je realizována Richardsonovou iterační metodou. Je představena analýza paměťových nároků paralelního výpočtu. Představujeme odvození cyklického rozdělení práce mezi procesory. Jsou představeny heuristické i metaheuristické algoritmy. Na závěr jsou řešení poskytnutá představenými algoritmy analyzována a vzájemně porovnána.

Klíčová slova: Metoda Hraničních Prvků, Hierarchické matice, paralelizace Richardsonovy metody, kompletní grafy, cyklické hranové ohodnocení kompletních grafů

Abstract

The subject of this thesis is the study and minimization of memory required by each processor during a parallel computation of the solution of a system of linear equations obtained by a collocational discretization of boundary integral equations. A solution of a system of differential equations using Boundary Element Method with collocation method of discretization is presented. To solve the underlying system of linear equations a parallel implementation of the Richardson iterative method is used. A so called cyclic task distribution is presented and developed. Heuristic and metaheuristic algorithms for solving the problem of cyclic task distribution are presented. At the end, all algorithms are compared with regards to the solution quality they are able to provide in a given time period.

Keywords: Boundary Elements Method, Hierarchical matrices, parallelization of Richardson method, complete graphs, cyclic edge labeling of complete graphs

Seznam použitých zkratek a symbolů

MHP	–	Metoda Hraničních Prvků,
MKP	–	Metoda Konečných Prvků,
$\bar{\Omega}$	–	uzávěr množiny Ω,
$C(\Omega)$	–	prostor spojitých reálných funkcí na oblasti Ω,
$C^i(\Omega)$	–	prostor reálných funkcí se spojitými derivacemi až do i-tého řádu na oblasti Ω,
$L^i(\Omega)$	–	prostor reálných funkcí takových, že $f \in L^p : \|f\| = \sqrt[p]{\int_{\mathbb{R}^n} f^p(x)\,dx} < \infty$ a f je měřitelná,
$\partial\Omega$	–	Hranice oblasti Ω,
Γ_D	–	Podmnožina $\partial\Omega$, na které platí Dirichletovy okrajové podmínky,
Γ_N	–	Podmnožina $\partial\Omega$, na které platí Neumannovy okrajové podmínky,
$\|\cdot\|$	–	Euklidovská norma,
$\mathcal{O}(\cdot)$	–	Asymptotická náročnost (výpočetní, paměťová),
$\lambda_{min}(A)$	–	Nejnižší vlastní číslo matice A,
$\lambda_{max}(A)$	–	Nejvyšší vlastní číslo matice A,
K_N	–	Kompletní graf na N vrcholech.

Obsah

Seznam tabulek

Seznam obrázků

Seznam algoritmů

1 Úvodní slovo

Při paralelním řešení soustav lineárních rovnic typu $Ax = b$ o stovkách miliónu neznámých, kdy A je hustá, je vhodné zabývat se způsoby redukce celkové paměti a meziprocesorové komunikace potřebné k realizaci paralelního výpočtu. V tomto textu je proto na modelovém příkladě diskretizace hraniční integrální rovnice ukázáno odvození husté soustavy $Ax = b$ a zamyšlení se nad partikulární třídou hierarchických reprezentací matice A. Vliv představené hierarchické reprezentace na celkové výpočetní a paměťové nároky na nalezení řešení $Ax = b$ je podrobně přezkoumán s tím závěrem, že má smysl zabývat se "chytřejšími" rozděleními práce mezi procesory. Poté je představena heuristická technika hledající optimální cyklické rozdělení práce mezi procesory, k jejímuž odvození byly využity poznatky z teorie grafů.

Tento text má následující strukturu:

1. **Kapitola 2**
   Nejdříve představíme teoretické základy potřebné k odvození hraničních integrálních rovnic, které si názorně ukážeme na dvou-dimenzionálním příkladu s Laplaceovým operátorem. Pro diskretizaci takto odvozených integrálních rovnic použijeme kolokační metodu, čímž odvodíme soustavu rovnic $Ax = b$, kde A je hustá matice. Rovněž představíme úvahy, které nám umožní reprezentovat A hierarchicky aniž bychom významně ovlivnili řešení soustavy $Ax = b$.

2. **Kapitola 3**
   Představíme paralelní iterační Richardsonovu metodu řešící $Ax = b$. Vzhledem k hierarchickému uložení matice A budeme zkoumat paměťové a výpočetní nároky nutné pro realizaci jedné iterace Richardsonovy metody a odvodíme motivaci k zamyšlení se nad způsoby, jak přiřadit práci mezi procesory tak, aby byl tyto nároky minimalizovány.

3. **Kapitola 4**
   Formulujeme problém nalezení rozdělení práce mezi procesory takového, které by minimalizovalo paměťové a výpočetní nároky nutné pro realizaci jedné Richardsonovy iterace. Motivujeme k hledání řešení v tzv. cyklickém tvaru a formulujeme heuristické a metaheuristické algoritmy, které jsou schopny nalézt cycklická rozdělení práce mezi procesory.

4. **Kapitola** 5
   Navrhnuté algoritmy poté analyzujeme a prezentujeme závěry vyjadřující se ke kvalitě nalezených řešení.

2 Úvod do problematiky metody hraničních prvků

2.1 Úvod

V této kapitole stručně představíme formulaci hraniční úlohy s odvozením metody hraničních prvků, její diskretizací a paralelizací.

Metoda hraničních prvků (MHP) je využívána při řešení soustav diferenciálních rovnic. Oproti metodě konečných prvků (MKP) má tu výhodu, že redukuje dimenzi problému, jelikož diskretizace úlohy neprobíhá uvnitř oblasti, ale jen na její hranici. Abychom však byli k MKP féroví, tak musíme zmínit, že MHP reprezentuje řešení v podobě husté matice. Pro úspěšné řešení problémů pomocí MHP je třeba znát tzv. fundamentální řešení diferenciálních operátorů popisujících problém, která jsou navíc v různých dimenzích odlišná. MHP vychází z věty o třech potenciálech a ze znalostí vlastností jednotlivých potenciálů. Na základě těchto znalostí pak MHP formuluje hraniční integrální rovnice popisující řešení v libovolném vnitřním bodě oblasti. Z těchto rovnic je pak dále odvozena metoda diskretizace, přičemž v této práci budeme uvažovat tzv. metodu kolokační.

Ukázkové odvození MHP bude předvedeno na dvou-dimenzionální úloze obsahující Laplaceův operátor. Následně bude představena paralelizace Jacobiho metody řešení soustavy rovnic. Použitá metoda paralelizace nám rovněž poskytne motivaci k zamyšlení se nad rozdělením práce mezi procesory, což je hlavním předmětem této práce a bude podrobněji rozebráno ve třetí kapitole.

2.2 Formulace okrajové úlohy

Definujme oblast $\emptyset \neq \Omega \subset \mathbb{R}^2$, přičemž budeme předpokládat, že Ω má dostatečně hladkou hranici. 2D - okrajovou úlohu se smíšenými podmínkami definujeme jako:

$$\begin{cases} -\triangle u & = f \quad \text{v } \Omega \\ u & = g \ \text{na } \Gamma_D \\ \frac{\partial u}{\partial n} & = h \ \text{na } \Gamma_N, \end{cases}$$

přičemž platí, že $\Gamma_N \cup \Gamma_D = \partial\Omega$, $\Gamma_N \cap \Gamma_D = \emptyset$ a o funkcích f, g a h předpokládáme, že jsou dostatečně hladké.

Pro vyřešení úlohy využijme následujících úvah, přičemž budeme předpokládat existenci funkcí $u, v \in C^2\left(\bar{\Omega}\right)$:

- S využitím první Greenovy věty odvodíme druhou Greenovu větu:

$$\int_\Omega -\triangle u \cdot v = \int_\Omega \nabla u \cdot \nabla v - \int_{\partial\Omega} \frac{\partial u}{\partial n} \cdot v \implies \int_\Omega \nabla u \cdot \nabla v = \int_\Omega -\triangle u \cdot v + \int_{\partial\Omega} \frac{\partial u}{\partial n} \cdot v,$$

$$\int_\Omega -\triangle v \cdot u = \int_\Omega \nabla v \cdot \nabla u - \int_{\partial\Omega} \frac{\partial v}{\partial n} \cdot u \implies \int_\Omega \nabla u \cdot \nabla v = \int_\Omega -\triangle v \cdot u + \int_{\partial\Omega} \frac{\partial v}{\partial n} \cdot u,$$

$$\implies \int_\Omega -\triangle u \cdot v + \int_{\partial\Omega} \frac{\partial u}{\partial n} \cdot v = \int_\Omega -\triangle v \cdot u + \int_{\partial\Omega} \frac{\partial v}{\partial n} \cdot u.$$

Definujme si tzv. fundamentální řešení.

Definice 2.1 *Funkci $v(y) = G(x,y) : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ nazveme fundamentálním řešením Laplaceovy rovnice ($\triangle u = 0$) ve dvou dimenzích právě tehdy, když $G(x,y) := \frac{1}{2\pi} \ln \frac{1}{\|x-y\|}$.*

Fundamentální řešení má následující vlastnosti (viz [6] a [1]):

- $G(x,y) \in C^\infty\left(\mathbb{R}^2 \times \mathbb{R}^2 \setminus \left\{(x,x) \in \mathbb{R}^2 \times \mathbb{R}^2\right\}\right)$.
- $G(x,x)$ je singulární.
- $G(x,y)$ je vzhledem k y harmonická funkce na $\mathbb{R}^2 \setminus \{x\}$.
- $-\triangle_y G(x,y) = \delta(x-y)$ ve smyslu distribucí, tedy $\forall \varphi \in C_0^\infty\left(\mathbb{R}^2\right) : \langle -\triangle_y G(x,y), \varphi\rangle = \langle \delta(x-y), \varphi\rangle = \varphi(x-y)$.

S využitím druhé Greenovy věty a fundamentálního řešení lze dokázat platnost tzv. věty o třech potenciálech.

Věta 2.1 *Uvažujme dříve definované Ω, fundamentální řešení $G(x,y)$ Laplaceovy rovnice ve dvou dimenzích a hledanou funkci $u \in C^2\left(\bar{\Omega}\right)$. Pak*

$$\forall x \in \Omega : \; u(x) = \int_\Omega f(y)\, G(x,y)\, dy + \int_{\partial\Omega} \frac{\partial u}{\partial n}(y)\, G(x,y)\, dl(y) - \int_{\partial\Omega} u(y)\, \frac{\partial G(x,y)}{\partial n(y)} dl(y).$$

Uvědomme si, že známe funkci $f(y)$, rovněž známe hodnotu $\frac{\partial u}{\partial n}$ na části hranice Γ_N (z Neumannových okrajových podmínek) a rovněž známe hodnotu $u(y)$ na části hranice Γ_D (z Dirichletových okrajových podmínek). Pro nalezení hodnoty $u(x)$ je tedy potřeba zjistit hodnotu $\frac{\partial u}{\partial n}$ na části hranice Γ_D a hodnotu $u(y)$ na části hranice Γ_N.

Pro zjednodušení zápisů následných odvození si definujme funkce:

$$\begin{array}{rcl} N\left(f\left(y\right)\right)\left(x\right) & := & \int_{\Omega} f\left(y\right) G\left(x,y\right) dy \\ \tilde{V}\left(\frac{\partial u}{\partial n}\left(y\right)\right)\left(x\right) & := & \int_{\partial\Omega} \frac{\partial u}{\partial n}\left(y\right) G\left(x,y\right) dl\left(y\right) \\ W\left(u\left(y\right)\right)\left(x\right) & := & \int_{\partial\Omega} u\left(y\right) \frac{\partial G(x,y)}{\partial n(y)} dl\left(y\right) \end{array}$$

Funkci $\tilde{V}\left(\frac{\partial u}{\partial n}\left(y\right)\right)\left(x\right) : \mathbb{R}^2 \to \mathbb{R}$ budeme nazývat **potenciálem jednoduché vrstvy**, $W\left(u\left(y\right)\right)\left(x\right) : \mathbb{R}^2 \to \mathbb{R}$ **potenciálem dvojvrstvy** a $N\left(f\left(y\right)\right)\left(x\right) : \mathbb{R}^2 \to \mathbb{R}$ **Newtonovým potenciálem**.

Vlastnost potenciálu jednoduché vrstvy:

Pro $\mu\left(y\right) := \frac{\partial u}{\partial n}\left(y\right) \in L^1\left(\partial\Omega\right) : \tilde{V}\left(\mu\left(y\right)\right)\left(x\right) \in C\left(\mathbb{R}^2\right)$ a $\triangle\tilde{V} = 0$.

Vlastnost potenciálu dvojvrstvy:

Pro $u\left(y\right) \in C\left(\partial\Omega\right) : W\left(u\left(y\right)\right)\left(x\right)$ je spojitá pouze na Ω a na $\mathbb{R}^2 \setminus \bar{\Omega}$. Pro $\forall x \in \partial\Omega$ obsahuje $W\left(u\left(y\right)\right)\left(x\right)$ skok.

Pro $\Omega \ni \tilde{x} \to x \in \partial\Omega : W\left(u\left(y\right)\right)\left(\tilde{x}\right) \to -\frac{1}{2}u\left(x\right) + \int_{\partial\Omega} u\left(y\right) \frac{\partial G(x,y)}{\partial n(y)} dl\left(y\right)$.

Vlastnost Newtonova potenciálu:

Pokud $f\left(y\right) \in L^{\infty}\left(\Omega\right)$, pak je $N\left(f\left(y\right)\right)\left(x\right) \in C^1\left(\mathbb{R}^2\right)$ (spojitý a spojitě diferencovatelný v $\mathbb{R}^2$).

Z výše uvedených vlastností potenciálů vyplývá:

$$\begin{array}{cccccccc} \tilde{x} \in \Omega : & u\left(\tilde{x}\right) & = & N\left(f\left(y\right)\right)\left(\tilde{x}\right) & + & \tilde{V}\left(\mu\left(y\right)\right)\left(\tilde{x}\right) & - & W\left(u\left(y\right)\right)\left(\tilde{x}\right) \\ \downarrow & \downarrow & & & & \downarrow & & \downarrow \\ x \in \partial\Omega : & u\left(x\right) & = & N\left(f\left(y\right)\right)\left(x\right) & + & V\left(\mu\left(y\right)\right)\left(x\right) & - & \left(-\frac{1}{2}u\left(x\right) + \int_{\partial\Omega} u\left(y\right) \frac{\partial G(x,y)}{\partial n(y)} dl\left(y\right)\right) \\ & u\left(x\right) & = & N\left(f\left(y\right)\right)\left(x\right) & + & \int_{\partial\Omega} \mu\left(y\right) G\left(x,y\right) dl\left(y\right) & - & \left(-\frac{1}{2}u\left(x\right) + \int_{\partial\Omega} u\left(y\right) \frac{\partial G(x,y)}{\partial n(y)} dl\left(y\right)\right) \end{array}$$

2.3 Diskretizace okrajové úlohy

Pro ukázku diskretizace a zformulování soustavy hraničních integrálních rovnic budeme uvažovat následující úlohu:

$$\begin{cases} -\triangle u & = 0 \ \text{ v}\,\Omega \\ u & = g \ \text{ na}\,\partial\Omega. \end{cases}$$

Ze zadání úlohy vyplývá:

- $\forall x \in \Omega : f(x) = 0 \Longrightarrow N(f(y))(x) = \int_\Omega f(y)\, G(x,y)\, dy = 0.$
- Jelikož zadání neobsahuje Neumannovy okrajové podmínky, pak je nutné dopočítat pouze $\mu(x) = \frac{\partial u}{\partial n}(x)$ na části hranice $\Gamma_D = \partial\Omega$.
- Známe hodnotu u na hranici, tzn., že známe $W(u(y))(x)$.

Budeme hledat řešení ve tvaru $u(\tilde{x}) = \tilde{V}(\mu(y))(\tilde{x}) - W(u(y))(\tilde{x})$ pro $\tilde{x} \in \Omega$. Z vlastností $\tilde{V}$ a W na hranici plyne:

$$\begin{array}{rcl} u(x) & = & \tilde{V}(\mu(y))(x) + \frac{1}{2}u(x) - W(u(y))(x) \\ u(x) & = & 2\tilde{V}(\mu(y))(x) - 2W(u(y))(x) \end{array}$$

Jelikož $u(x \in \partial\Omega) = g(x) \Longrightarrow g(x) = 2\tilde{V}(\mu(y))(x) - 2W(g(y))(x)$, budeme hledat řešení hraniční integrální rovnice $\tilde{V}(\mu(y))(x) = \frac{1}{2}g(x) + W(g(y))(x)$ na $\partial\Omega$, a sice: $\forall x \in \partial\Omega : \int_{\partial\Omega} \mu(y)\, \frac{1}{2\pi} \ln \frac{1}{\|x-y\|} dl(y)$

$$\forall x \in \partial\Omega : \int_{\partial\Omega} \mu(y)\, \frac{1}{2\pi} \ln \frac{1}{\|x-y\|} dl(y) \quad = \quad \tfrac{1}{2}g(x) + \int_{\partial\Omega} g(y)\, \frac{\partial\left(\frac{1}{2\pi}\ln\frac{1}{\|x-y\|}\right)}{\partial n(y)} dl(y) \quad \heartsuit$$

Pro diskretizaci řešení $\heartsuit$ budeme uvažovat tzv. kolokační metodu. Ta spočívá v rozdělení $\partial\Omega$ na n úseček $s_i, i \in \{1,\ldots,n\}$ takových, že $\bigcup_{i=1}^{n} \bar{s_i} = \partial\Omega$ a $i,j \in \{1,\ldots,n\} : i \neq j, s_i \cap s_j = \emptyset$ (úsečky neuvažujeme jako přímou spojnici dvou bodů, ale jako část křivky). Pro aproximaci řešení budeme uvažovat po částech konstantní bázové funkce $e_i(x)$ takové, že $e_i(x)\,|_{s_j} = \delta_{ij}$ pro $i,j \in \{1,\ldots,n\}$. Hustotu potenciálu jednoduché vrstvy budeme hledat v lineárním obalu těchto bázových funkcí, tedy:

$$\mu(y) = \frac{\partial u}{\partial n}(y) \approx \sum_{j=1}^{n} \alpha_j e_j(y).$$

Označme si střed úsečky s_i jako x_i. **Kolokační metoda vyžaduje splnění rovnice ♡ pouze v bodech $\mathbf{x_i}$.**

2.4 Sestavení soustavy rovnic

Naším úkolem je nyní nalézt aproximaci $\mu(y)$ v podobě $\sum_{j=1}^{n} \alpha_j e_j(y)$, sestavme si proto n lineárních rovnic o n neznámých v podobě $A\alpha = b$, kde A, α a b odvodíme na základě úprav ♡ níže.

$$
\begin{aligned}
\forall x \in \partial\Omega: \quad & \int_{\partial\Omega} \mu(y) \frac{1}{2\pi} \ln \frac{1}{\|x-y\|} dl(y) &&= \tfrac{1}{2} g(x) + \int_{\partial\Omega} g(y) \frac{\partial\left(\frac{1}{2\pi} \ln \frac{1}{\|x-y\|}\right)}{\partial n(y)} dl(y) \\
\forall x \in \partial\Omega: \quad & \int_{\partial\Omega} \sum_{j=1}^{n} \alpha_j e_j(y) \frac{1}{2\pi} \ln \frac{1}{\|x-y\|} dl(y) &&= \tfrac{1}{2} g(x) + \int_{\partial\Omega} g(y) \frac{\partial\left(\frac{1}{2\pi} \ln \frac{1}{\|x-y\|}\right)}{\partial n(y)} dl(y) \\
\forall x \in \partial\Omega: \quad & \sum_{j=1}^{n} \alpha_j \int_{s_j} \frac{1}{2\pi} \ln \frac{1}{\|x-y\|} dl(y) &&= \tfrac{1}{2} g(x) + \int_{\partial\Omega} g(y) \frac{\partial\left(\frac{1}{2\pi} \ln \frac{1}{\|x-y\|}\right)}{\partial n(y)} dl(y) \\
\forall x_i: \quad & \left(\sum_{j=1}^{n} \alpha_j \int_{s_j} \frac{1}{2\pi} \ln \frac{1}{\|x_i-y\|} dl(y)\right)(x_i) &&= \tfrac{1}{2} g(x_i) + \left(\int_{\partial\Omega} g(y) \frac{\partial\left(\frac{1}{2\pi} \ln \frac{1}{\|x_i-y\|}\right)}{\partial n(y)} dl(y)\right)(x_i).
\end{aligned}
$$

$\bar{\alpha}$ tedy reprezentuje vektor neznámých multiplikátorů bázových funkcí, b reprezentuje vektor známých hodnot z pravé strany výrazu ♡, přičemž $b_i = \frac{1}{2} g(x_i) + \left(\int_{\partial\Omega} g(y) \frac{\partial\left(\frac{1}{2\pi} \ln \frac{1}{\|x_i-y\|}\right)}{\partial n(y)} dl(y)\right)(x_i)$ a A je matice soustavy, kdy $A_{ij} = \left(\int_{s_j} \frac{1}{2\pi} \ln \frac{1}{\|x_i-y\|} dl(y)\right)(x_i)$(podrobnější informace o kolokační diskretizační technice je možno nalézt v [4]). Matice A zkonstruovaná na základě výše popsaného odvození je hustá, tzn. že obsahuje řádově $\mathcal{O}(n^2)$ nenulových hodnot a je tedy třeba zamyslet se nad její efektivnější reprezentací, viz následující část textu.

Všimněme si, že hodnota A_{ij} popisuje vztah mezi úsečkou s_i a úsečkou s_j. Předpokládejme, že čím vzdálenější jsou s_i a s_j, tím nižší vliv bude mít x_i na hodnotu integrálu přes s_j et vice versa. Předpokládáme tedy, že některé prvky matice A_{ij} můžeme aproximovat aniž by to významně ovlivnilo hodnotu nalezeného řešení. Pro úplnost si uveďme kritéria, která nám pomohou rozhodnout zdali má být vazba mezi úsečkami s_i a s_j počítána přesně či nikoliv.

Mějme úsečku s_i, jejím diametrem budeme rozumět funkci:

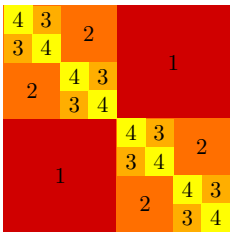
$$\operatorname{diam}(s_i) = \min\left\{ r : \text{ existuje kruh } k_r \subseteq \mathbb{R}^2 \text{o poloměru } r \text{takový, že } \bar{s_i} \subseteq k_r \right\}.$$

Dále mějme úsečku s_j, pak vzdálenost úseček s_i a s_j definujeme jako funkci:

$$\text{dist}\left(s_i, s_j\right) = \min\left\{\|x - y\| : x \in \bar{s_i}, y \in \bar{s_j}\right\}.$$

O úsečkách s_i a s_j pak řekneme, že jsou si **blízké** právě tehdy, když $\min\left\{\text{diam}\left(s_i\right), \text{diam}\left(s_j\right)\right\} \geq \text{dist}\left(s_i, s_j\right)$ a hodnoty A_{ij}, A_{ji} aproximovat nebudeme. V opačném případě budeme považovat úsečky s_i a s_j za **vzdálené** a hodnoty A_{ij}, A_{ji} budeme aproximovat.

Nyní si definujme tzv. hierarchickou matici $\mathcal{H} \in \mathbb{R}^{n \times n}$ takovou, že $\mathcal{H} \approx A$ (detaily v [5]). Pro sestrojení $\mathcal{H}$ je třeba rozdělit A do vzájemně disjunktních souvislých bloků, označme si jeden takový blok jako $A \supseteq C \in \mathbb{R}^{m \times m}$. Hodnoty C máme možnost aproximovat například tak, že $C \approx UV^T$, kde $U, V \in \mathbb{R}^{m \times k}$ a $1 \leq k < \frac{m}{2}$ je parametr ovlivňující míru aproximace. Matice $\mathcal{H}$ pak bude na pozicích indukovaných podmaticí C obsahovat hodnoty aproximované součinem UV^T, přičemž nebudeme ukládat produkt součinu samotného, ale matice rozkladu U a V. Ukázka hierarchické aproximace A je znázorněna na obrázku 1.



$\mathcal{H}$

Obrázek 1: Ukázka čtyř-úrovňové hierarchické aproximace matice A pro $n = 2^k \geq 16$. Hodnoty ve vybarvených blocích vyjadřují míru dat potřebných k jejich uložení, 1-nejnižší, 4-nejvyšší.

Příklad hierarchické aproximace A na obrázku 1 náleží třídě aproximací s následujícími vlastnostmi (♠):

- $n = 2^k$.
- $\forall l \in \{1, \ldots, k-1\}$ matice A obsahuje $m = 2^l$ podmatic C^l, kde $\mathbb{R}^{\frac{n}{m} \times \frac{n}{m}} \ni C^l \approx UV^T$, přičemž $U, V \in \mathbb{R}^{\frac{n}{m} \times l}$.
- pro $l = k$ matice A obsahuje $m = 2^{l+1}$ podmatic C^l, kde $\mathbb{R} \ni C^l$.
- asymptotická náročnost uložení jedné matice C^l je $\mathcal{O}\left(2^{k-l} l\right)$.
- matic C^l je 2^l, uložení všech C^l tedy vyžaduje $\mathcal{O}\left(2^k l\right)$ jednotek paměti.

- uložení celé $\mathcal{H}$ tedy vyžaduje $\mathcal{O}\left(\sum_{l=1}^{\log_2 n} nl\right) = \mathcal{O}\left(n\sum_{l=1}^{\log_2 n} l\right) = \mathcal{O}\left(n\frac{(1+\log_2 n)\log_2 n}{2}\right) = \mathcal{O}\left(n\log_2^2 n\right)$ jednotek paměti.
- všimněme si, že $\mathcal{H}$ je hustá kolem diagonály.

V analýze paměťových a výpočetních nároků paralelního výpočtu v následující části textu budeme uvažovat soustavu $\mathcal{H}x = b$, kde $\mathcal{H}$ splňuje ♠.

3 Paralelizace výpočtu řešení soustavy získané MHP

Pro paralelizaci řešení soustav rovnic $\mathcal{H}x = b$, kde $\mathcal{H}$ je hierarchická matice, byla zvolena Richardsonova iterační metoda. V této části textu si připomeneme algoritmus Richardsonovy metody, navrhneme jeho paralelizaci a zamyslíme se nad vlastnostmi navrhntutých paralelizačních technik.

Richardsonova metoda je používána k iteračnímu řešení soustav lineárních rovnic typu $Ax = b$, přičemž je nutné vhodně zvolit parametr ω takový, aby výraz $x^{k+1} = x^k + \omega\left(b - Ax^k\right)$ konvergoval (optimální hodnota je $\omega = \frac{2}{\lambda_{min}(A)+\lambda_{max}(A)}$).

Algoritmus Richardsonovy iterační metody definujme následovně:

Vstupy : matice A, vektor pravých stran b, tolerance $\varepsilon > 0$, konvergenční parametr ω
Výstupy : aproximace řešení $\hat{x}$

Inicializace: $x^0 \in \{0\}^n$, $k = 0$

begin
  while $\left\|Ax^k - b\right\| > \varepsilon$ **do**
    $x^{k+1} = x^k + \omega\left(b - Ax^k\right)$
    $k = k + 1$
  end
  $\hat{x} = x^k$
  return $\hat{x}$
end

Algoritmus 1: Algoritmus Richardsonovy iterační metody.

Všimněme si, že jádrem iterace algoritmu 1 je výraz Ax^k a vzpomeňme si, že:

$$w = Ax^k = \begin{bmatrix} \sum_{j=1}^{n} A_{1j}x_j \\ \sum_{j=1}^{n} A_{2j}x_j \\ \vdots \\ \sum_{j=1}^{n} A_{nj}x_j \end{bmatrix},$$

tzn. že pro libovolné $i, j \in \{1, \ldots, n\}$ se hodnota A_{ij} ve výpočtu vektoru w použije pouze jedenkrát. Pro paralelizaci výpočtu w můžeme tedy předem definovat, které procesory budou potřebovat jaké hodnoty A_{ij}, aniž bychom cokoliv počítali redundatně. Rovněž si uvědomme, že matice A je v tomto kontextu hierarchická, tzn. že obsahuje některé bloky, které jsou ve formě součinu dvou vektorů. Buď

$C = UV^T$ jedním z těchto bloků, přičemž $U, V \in \mathbb{R}^{m \times k}$. Prvky matice A indukované blokem C se podílí na částečné hodnotě vektoru w, označme si jej pro přehlednost jako $w_C = Cx_C$, kde $x_C \subseteq x$ je část vektoru x potřebná k realizaci části výpočtu Ax^k indukované blokem C.

Vektor w_C můžeme spočítat například tak, že $w_C = \left(UV^T\right) x_C$, což vyžaduje $\mathcal{O}\left(km^2\right)$ operací pro výpočet UV^T a $\mathcal{O}\left(m^2\right)$ operací pro výpočet Cx_C. Rovněž jej však můžeme spočítat tak, že $w_C = U\left(V^T x_C\right)$, což vyžaduje $\mathcal{O}\left(km\right)$ operací pro výpočet $V^T x_C$ a $\mathcal{O}\left(km\right)$ operací pro výpočet $U\left(V^T x_C\right)$. Paralelizaci součinu Ax^k tedy budeme realizovat tak, že $w_C = U\left(V^T x_C\right)$.

3.1 Definice a analýza paralelního algoritmu Richardsonovy metody

Pro názornost budeme předpokládat, že $n = 2^k$ a počet procesorů $N = 2^l$, přičemž $1 \leq N \leq n$. Budeme rovněž uvažovat hierarchickou matici A s vlastnostmi splňující ♠. Pro snadnější implementaci paralelizace rozdělíme matici A na N^2 vzájemně disjunktních bloků $B_{pq} \in \mathbb{R}^{2^{(k-l)} \times 2^{(k-l)}}$, kde B_{pq} reprezentuje hodnoty matice A na řádcích $\frac{n}{N}(p-1)+1$ až $\frac{n}{N}p$ a ve sloupcích $\frac{n}{N}(q-1)+1$ až $\frac{n}{N}q$. Pro názornost je rozdělení matice A do bloků B_{pq} uvedeno na obrázku 2.

B_{11}	B_{12}	B_{13}	B_{14}
B_{21}	B_{22}	B_{23}	B_{24}
B_{31}	B_{32}	B_{33}	B_{34}
B_{41}	B_{42}	B_{43}	B_{44}

A

Obrázek 2: Ukázka rozdělení matice A do bloků B_{pq}, kdy $n = 16$ a $N = 4$.

Definujme si přiřazení práce procesorům jako $R \in \{1, \ldots, N\}^{N \times N}$, přičemž zápisem $R_{ij} = m$ budeme rozumět to, že procesor m bude počítat část součinu Ax^k indukovanou blokem B_{ij}, rovněž budeme požadovat aby $R_{ii} = i$ (z vlastností ♠ a hustoty hierarchické matice kolem diagonály).

Pro paralelizaci algoritmu 1 můžeme využít například modelu **Master-Slave**, kde jeden z procesorů převezme úlohu organizátora dělby práce (Master), kterou bude rozdělovat mezi zbývající procesory (Slaves), přičemž budeme předpokládat, že všechny procesory jsou totožně výkonné. Definujme tedy

paralelní algoritmus Richardsonovy metody vyplývající z předpokladu, že právě jeden procesor bude počítat celou část výpočtu Ax^k indukovanou nějakým blokem B_{pq}.

Vstupy : matice A, vektor pravých stran b, tolerance $\varepsilon > 0$, konvergenční parametr ω, rozdělení práce R

Výstupy : aproximace řešení $\hat{x}$

Inicializace: $x^0 \in \{0\}^n$, $k = 0$

begin

Master rozešle každému procesoru m data reprezentující takové bloky B_{ij}, že $R_{ij} = m$

while $\left\|Ax^k - b\right\| > \varepsilon$ **do**

Master rozešle x^k mezi procesory tak, aby každý procesor m měl jen tu část x^k, kterou potřebuje k výpočtu části Ax^k indukované bloky B_{ij} takovými, že $R_{ij} = m$

Každý procesor spočítá svou část součinu Ax^k

Procesory zašlou částečné hodnoty součinu Ax^k zpět Masterovi, jenž vytvoří vektor Ax^k a provede výpočet $x^{k+1} = x^k + \omega\left(b - Ax^k\right)$

$k = k + 1$

end

$\hat{x} = x^k$

return $\hat{x}$

end

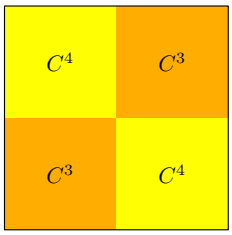
Algoritmus 2: Paralelní algoritmus Richardsonovy iterační metody. Tento algoritmus běží na procesoru označeného jako Master.

V analýze algoritmu 2 budeme používat následující konstanty:

- t_c bude reprezentovat dobu přenosu jedné konstanty mezi dvěma procesory.
- t_a bude reprezentovat dobu potřebnou na sečtení dvou konstant jedním procesorem.
- t_m bude reprezentovat dobu potřebnou k vynásobení dvou konstant jedním procesorem, přičemž můžeme předpokládat, že $t_m = \gamma t_a$ pro nějaké $0 < \gamma \in \mathbb{R}$.

Podívejme se na vztah mezi bloky hierarchické matice,$C^\gamma \in \mathbb{R}^{2^{k-\gamma}\times 2^{k-\gamma}}$, $\gamma \in \{1,\ldots,k-1\}$ a bloky $B_{ij} \in \mathbb{R}^{2^{k-l}\times 2^{k-l}}$:

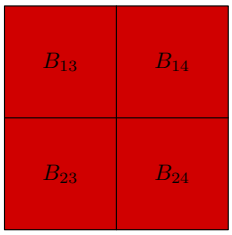
- Pokud $2^{k-l} \geq 2^{k-\gamma}$, pak je blok C^γ plně obsažen v některém bloku B_{ij} (vyplývá z předpokladu, že rozměry B_{ij} i C^γ jsou mocninou dvojky). Jeden procesor je tedy jednoduše schopen spočítat část Ax^k indukovanou blokem C^γ. Paměť potřebná pro realizaci části výpočtu Ax^k indukovanou blokem C^γ je $\mathcal{O}\left(2^{k-\gamma}\gamma\right)$, potřebný výpočetní čas je pak $\mathcal{O}\left(2^{k-\gamma}\gamma\left(t_a+t_c\right)\right)$.



B_{11}

Obrázek 3: Znázornění vztahu mezi bloky C^γ a blokem B_{ij}, kdy $2^{k-l} \geq 2^{k-\gamma}$ pro konkrétní blok B_{11}, přičemž $n = 16$ a $N = 4$.

- Pokud $2^{k-l} < 2^{k-\gamma}$, tak jsme schopni blok C^γ rozdělit do $2^{2(l-\gamma)}$ vzájemně disjunktních bloků B_{ij} (vyplývá z předpokladu, že rozměry B_{ij} i C^γ jsou mocninou dvojky, $\frac{2^{k-\gamma}}{2^{k-l}} = 2^{l-\gamma}$). Pro výpočet části Ax^k indukované jedním takovým blokem B_{ij} bude zapotřebí $\mathcal{O}\left(2^{k-l}\gamma\right)$ paměti a výpočetního času, přičemž potřebná data získáme jedoduše příslušným rozdělením vektorů u a v ($C^\gamma = uv^T$, $u, v \in \mathbb{R}^{2^{k-\gamma}\times\gamma}$). V závislosti na R pak bude pro výpočet Ax^k indukované C^γ zapotřebí od $\mathcal{O}\left(2^{k-\gamma}\gamma\right)$ do$\mathcal{O}\left(2^{k-l}\gamma \cdot 2^{2(l-\gamma)}\right) = \mathcal{O}\left(\gamma \cdot 2^{k+l-2\gamma}\right)$ paměti a výpočetního času.



C^1

Obrázek 4: Znázornění vztahu mezi blokem C^γ a bloky B_{ij}, kdy $2^{k-l} < 2^{k-\gamma}$ pro konkrétní blok C^1, přičemž $n = 16$ a $N = 4$.

Nyní se zamysleme nad vlastnostmi algoritmu 2, přičemž budeme předpokládat, že se vykoná takový počet iterací, že čas potřebný na zaslání dat reprezentující bloky B_{ij} jednotlivým procesorům bude asymptoticky nevýznamný. Jednotlivé složky algoritmu jsou:

- Master rozešle x^k mezi procesory tak, aby každý procesor m obdržel jen tu část x^k, kterou potřebuje k výpočtu části Ax^k indukované bloky B_{ij} takovými, že $R_{ij} = m$.
  Všimněme si, že libovolný prvek vektoru x^k může být ve výpočtu využit všemi procesory. Buď $X_i := \{R_{ji}, j \in \{1, \ldots, N\}\}$ množina procesorů, které k výpočtu své části Ax^k potřebují hodnotu x^k_p, kde $(i-1)\frac{n}{N} < p \leq i\frac{n}{N}$, pak dobu potřebnou pro zaslání x^k_p procesorům $m \in X_i$ lze realizovat v čase $\mathcal{O}\left(t_c \log_2 |X_i|\right)$(viz [2]), celý vektor x^k lze pak rozeslat mezi procesory v čase $\mathcal{O}\left(t_c \frac{n}{N} \sum_{i=1}^{N} \log_2 |X_i|\right)$. Buď $O_m := \{i : \exists j : R_{ji} = m\}$, pak na procesoru m budeme potřebovat $\mathcal{O}\left(\frac{n}{N} |O_m|\right)$ paměti k uložení té části x^k potřebné k výpočtu Ax^k přidělené procesoru m.
- Každý procesor spočítá svou část součinu Ax^k.
  - Výpočetní čas
    * Pro $\gamma \in \{1, \ldots, l-1\}$ musíme část výpočtu indukovanou blokem C^γ dále dělit, jelikož $2^{k-l} < 2^{k-\gamma}$. Ukázali jsme si, že jsme schopni C^γ rozdělit na $2^{2(l-\gamma)}$ bloků B_{ij}. Z ♠ víme, že počet bloků C^γ je 2^γ. Pro výpočet části řešení Ax^k indukované bloky $C^{\gamma<l}$ tedy budeme potřebovat $\sum_{\gamma=1}^{l-1} 2^{2(l-\gamma)} 2^\gamma$ bloků B_{ij}, přičemž výpočetní náročnost výpočtu části Ax^k indukované bloky $C^{\gamma<l}$ bude v rozmezí od $\mathcal{O}\left(\sum_{\gamma=1}^{l-1} 2^{k-\gamma} \gamma \left(t_a + t_m\right)\right) = \mathcal{O}\left(\left(t_a + t_m\right) n \log_2 N\right)$ do $\mathcal{O}\left(\sum_{\gamma=1}^{l-1} \gamma \cdot 2^{k+l-2\gamma} \left(t_a + t_m\right)\right) = \mathcal{O}\left(\left(t_a + t_m\right) nN \log_2 N\right)$ času.
      Jelikož N vyjadřuje počet procesorů, tak v ideálním případě paralelní implementace budeme schopni tuto část řešení vypočítat v čase od $\mathcal{O}\left(\left(t_a + t_m\right) \frac{n}{N} \log_2 N\right)$ do $\mathcal{O}\left(\left(t_a + t_m\right) n \log_2 N\right)$.
    * Naopak pro $\gamma \in \{l, \ldots, k\}$ jsme schopni výpočet jednoduše rozdělit, jelikož blok C^γ je plně obsažen v nějakém bloku B_{ij}. Časová náročnost výpočtu části Ax^k indukované bloky C^γ pro $\gamma \in \{l, \ldots, k\}$ je tedy $\mathcal{O}\left(\sum_{\gamma=l}^{k} 2^{k-\gamma} \gamma 2^\gamma \left(t_a + t_m\right)\right) = \mathcal{O}\left(2^k \sum_{\gamma=l}^{k} \gamma \left(t_a + t_m\right)\right) = \mathcal{O}\left(\left(t_a + t_m\right) n \log_2\left(\frac{n}{N}\right) \log_2 (nN)\right)$. Pokud se nám podaří rozdělit tuto část výpočtu Ax^k mezi procesory rovnoměrně, pak dosáhneme časové náročnosti $\mathcal{O}\left(\left(t_a + t_m\right) \frac{n}{N} \log_2\left(\frac{n}{N}\right) \log_2 (nN)\right)$.

* Realizaci paralelního výpočtu Ax^k jsme tedy schopni v ideálním případě vykonat v čase od $\mathcal{O}\left((t_a + t_m)\,\frac{n}{N}\left(\log_2\left(\frac{n}{N}\right)\log_2(nN) + \log_2 N\right)\right)$ do $\mathcal{O}\left((t_a + t_m)\left(\frac{n}{N}\log_2\left(\frac{n}{N}\right)\log_2(nN) + n\log_2 N\right)\right)$.

– Paměťové nároky

* Odhadněme pamětový rozsah, který budeme potřebovat pro realizaci výpočtu. Data potřebná pro výpočet Ax^k indukované pouze blokem $C^\gamma = UV^T$ vyžadují $\mathcal{O}\left(2^{k-\gamma}\gamma\right)$ paměti.
Pro $l \leq \gamma \leq k$ nebudeme C^γ dělit mezi bloky B_{ij}, a proto bude náročnost jejich uložení řádově $\mathcal{O}\left(\sum_{\gamma=l}^{k} 2^{k-\gamma}\gamma 2^\gamma\right) = \mathcal{O}\left(n\log_2\left(\frac{n}{N}\right)\log_2(nN)\right)$ jednotek paměti.
Pro $1 \leq \gamma < l$ budeme C^γ dělit mezi bloky B_{ij}. Minimální paměť potřebná k uložení dat reprezentujících C^γ je řádově $\mathcal{O}\left(\sum_{\gamma=1}^{l-1} 2^{k-\gamma}\gamma 2^\gamma\right) = \mathcal{O}\left(n\log_2^2 N\right)$, maximální potřebná paměť potřebná k uložení dat reprezentujících C^γ je pak řádově $\mathcal{O}\left(\sum_{\gamma=1}^{l-1} 2^{2(l-\gamma)}2^\gamma 2^{k-l}\gamma\right) = \mathcal{O}\left(nN\log_2 N\right)$.
Paměťpotřebná pro realizaci výpočtu bude tedy v rozmezí od $\mathcal{O}\left(n\log_2\left(\frac{n}{N}\right)\log_2(nN) + n\log_2^2 n\right)$ do $\mathcal{O}\left(nN\log_2 N\right)$.

- Procesory zašlou částečné hodnoty součinu Ax^k zpět Masterovi.
Buď $I_m := \{i : \exists j : R_{ij} = m\}$, pak procesor m vyžaduje řádově $\mathcal{O}\left(t_c\frac{n}{N}|I_m|\right)$ paměti k uložení svého dílu částečného řešení Ax^k. Zaslání všech částečných součinů zpět Masterovi bude vyžadovat $\mathcal{O}\left(t_c\frac{n}{N}\sum_{m=1}^{N}|I_m|\right)$ času, jelikož $1 \leq |I_m| \leq N$, pak tento krok algoritmu bude vyžadovat od $\mathcal{O}\left(t_c n\right)$ do $\mathcal{O}\left(t_c nN\right)$ jednotek času.

Potřebný čas na realizaci jedné iterace bude tedy:

$$\text{min:}\quad \mathcal{O}\left(t_c\frac{n}{N}\sum_{i=1}^{N}\log_2|X_i| + t_c\frac{n}{N}\sum_{m=1}^{N}(|O_m| + |I_m|) + (t_a + t_m)\,\frac{n}{N}\left(\log_2\left(\frac{n}{N}\right)\log_2(nN) + \log_2 N\right)\right)$$

$$\text{max:}\quad \mathcal{O}\left(t_c\frac{n}{N}\sum_{i=1}^{N}\log_2|X_i| + t_c\frac{n}{N}\sum_{m=1}^{N}(|O_m| + |I_m|) + (t_a + t_m)\left(\frac{n}{N}\log_2\left(\frac{n}{N}\right)\log_2(nN) + n\log_2 N\right)\right)$$

Potřebná paměť k realizaci jedné iterace na procesoru m bude tedy:

Úkolem bude tedy definovat R tak, aby byly paměťové a časové nároky na výpočet minimalizovány, přičemž v této práci se zaměříme na minimalizaci nároků algoritmu závislých na I_m a O_m.

$$\begin{aligned}\text{min:}\quad & \mathcal{O}\left(\frac{n}{N}\left(|O_m|+|I_m|\right)+\frac{n}{N}\left(\log_2\left(\frac{n}{N}\right)\log_2\left(nN\right)+\log_2 N\right)\right)\\ \text{max:}\quad & \mathcal{O}\left(\frac{n}{N}\left(|O_m|+|I_m|\right)+n\log_2 N\right)\end{aligned}$$

Všimněme si, že existuje vztah mezi množinami I_m a O_m:

- Předpokládejme, že procesoru m přiřadíme bloky B_{1j} pro $j \in \{1,\ldots,N\}$, pak $O_m = \{1\}$ a $I_m = \{1,\ldots,N\}$ a čas potřebný pro zaslání příslušné části vektoru x^k procesoru m je $\mathcal{O}\left(t_c\frac{n}{N}\right)$ a pro jeho uložení je třeba $\mathcal{O}\left(\frac{n}{N}\right)$ paměti, nicméně je třeba $\mathcal{O}\left(n\right)$ paměti pro uložení částečného vektoru Ax^k a $\mathcal{O}\left(t_c n\right)$ času pro jeho zaslání zpět Masterovi.
- Oproti tomu předpokládejme, že procesoru m přiřadíme bloky B_{ij} pro $i,j \in \left\{1,\ldots,\sqrt{N}\right\}$, pak $O_m = I_m = \left\{1,\ldots,\sqrt{N}\right\}$ a čas potřebný pro zaslání příslušné části vektoru x^k procesoru m je $\mathcal{O}\left(t_c\frac{n}{\sqrt{N}}\right)$ a pro jeho uložení je třeba $\mathcal{O}\left(\frac{n}{\sqrt{N}}\right)$ paměti, rovněž je třeba $\mathcal{O}\left(\frac{n}{\sqrt{N}}\right)$ paměti pro uložení částečného vektoru Ax^k a $\mathcal{O}\left(t_c\frac{n}{\sqrt{N}}\right)$ času pro jeho zaslání zpět Masterovi.
- Vidíme tedy, že v závislosti na R můžeme komunikaci mezi všemi procesory provádět v čase od $\mathcal{O}\left(t_c n\sqrt{N}\right)$ do $\mathcal{O}\left(t_c nN\right)$, potřebná data pak ukládat s využitím od $\mathcal{O}\left(\frac{n}{\sqrt{N}}\right)$ do $\mathcal{O}\left(n\right)$ paměti na jednom procesoru. (Ne)Vhodnou definicí R jsme tedy schopni změnit paměťové nároky realizace výpočtu závislých na I_m a O_m až o faktor $\sqrt{N}$.

3.2 Závěr kapitoly

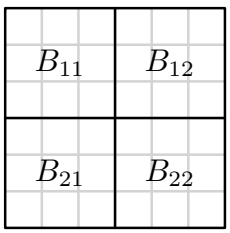
Stručně jsme představili odvození, diskretizaci, paralelizaci Metody Hraničních prvků a ukázali jsme motivaci pro netriviální distribuci práce mezi procesory. V následující kapitole odvodíme způsoby, jak práci procesorům přiřadit tak, aby byla paměť potřebná k výpočtu závislá na množinách I_m a O_m snížena na co nejnižší hodnotu.

4 Problematika rozdělení práce mezi procesory

4.1 Úvod

V předchozích kapitolách bylo popsáno odvození řešení hraniční úlohy společně s jednou z metod diskretizace a uložení soustavy reprezentující řešení úlohy. Ukázali jsme si, že má smysl hledat rozdělení práce mezi procesory takové, aby byla minimalizována komunikace mezi procesory a aby byly sníženy paměťové nároky potřebné k realizaci výpočtu na každém procesoru. V této kapitole si tedy ukážeme metody rozdělení práce mezi procesory tak, aby byla výpočetní zátež rozložena rovnoměrně a aby byla minimalizována využitá paměť jednotlivými procesory. Nejdříve se seznámíme s úvahami jak optimálního rozdělení práce dosáhnout obecnými metodami, které jsou však výpočetně neefiktivní. Proto navážeme poznatky a nápady vyplývající z teorie grafů, které nám umožní naleznout paměťově efektivní rozdělení práce v rozumném čase.

Diskretizací hraniční úlohy v předchozí kapitole jsme obdrželi matici $A \in \mathbb{R}^{n \times n}$, kde $n \in \mathbb{N}$ vyjadřuje počet diskretizačních prvků na hranici úlohy. Buď $N \in \mathbb{N}$ počet procesorů, na kterých chceme úlohu řešit. Pro snížení náročnosti hledání efektivnějších způsobů rozdělení práce budeme na A nahlížet jako na matici obsahující N^2 vzájemně disjunktních čtvercových podmatic B_{ij} $(B_{ij} \subseteq A) \in \mathbb{R}^{m \times m}$, kde $m \in \mathbb{N}$, $m \cdot N = n$ a $i, j \in \{1, \ldots, N\}$. Zápisem B_{ij} budeme rozumět matici obsahující prvky i-té m-tice řádků a j-té m-tice sloupců matice A.



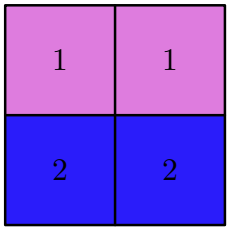
A

Obrázek 5: Rozdělení matice $A \in \mathbb{R}^{6 \times 6}$ do čtyř čtvercových bloků B_{ij}.

Dále si definujme datovou kolekci D_i pro $i \in \{1, \ldots, N\}$. Pro výpočet části řešení indukované podmaticí B_{ij} pak budeme potřebovat datové bloky D_i a D_j.

Označme si jednotlivé procesory indexy $l \in \{1, \ldots, N\}$. Dále si označme libovolné přiřazení podmatic B_{ij} procesorům jako $R \in \{1, \ldots, N\}^{N \times N}$, kde hodnota R_{ij} označuje index procesoru, kterému je

přiřazena podmatice B_{ij}. Pro lepší pochopení přiřazení bloků B_{ij} procesorům je znázorněn příklad na obrázku 6.

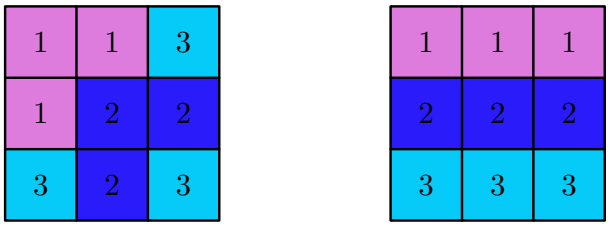


R

Obrázek 6: Ukázkové přiřazení bloků matice A procesorům s indexy 1 a 2.

Definice 4.1 *Paměťovou náročnost přiřazení bloků B_{ij} procesorům definujme jako cenovou funkci $c: R \to \mathbb{N}$, kde $c(R) = \max_{l \in \{1,\ldots,N\}} \{|\{i: R_{ij} = l \vee R_{ji} = l, \forall i,j \in \{1,\ldots,N\}\}|\}$, tedy maximální počet prvků datové kolekce potřebné k výpočtu všech řešení indukovaných podmaticemi B_{ij} přiřazených procesoru s libovolným s indexem. Zaměřme se nyní na nalezení takového R, aby největší využití paměti libovolným procesorem bylo co nejmenší. Buď $L = \{1,\ldots,N\}$, zápisem R^{opti} budeme rozumět nejlepší přiřazení R, neboli*

$$\forall R \in L^{N \times N}: c\left(R^{opti}\right) \leq c(R).$$



R S

Obrázek 7: Příklad dvou přiřazení R a S, kde $N = 3$ a $c(R) < c(S)$.

4.2 Řešení hrubou silou

Prvotním přístupem, který by zaručeně produkoval nejlepší výsledky pro libovolný počet procesorů, je nalezení R_{opti} hrubou silou, tzn., že bychom ze všech permutací R zvolili tu nejlepší. Na libovolné R můžeme v tomto kontextu nahlížet jako na prvek, do kterého máme možnost přiřadit N^2 hodnot z množiny $\{1, \ldots, N\}$. Jelikož máme předepsány hodnoty R_{ii}(viz definice 4.2), tak různých možností jak tímto způsobem vygenerovat R je $N^{N(N-1)}$, čehož asymptotická náročnost je $\mathcal{O}\left(N^{N^2}\right)$.

Definice 4.2 *Přiřazení R takové, že každému procesoru přiřadí totožný počet podmatic B_{ij}, budeme nazývat přiřazením rovnoměrným.*

Matice A je obecně hustá, přičemž podmatice B_{ii} kolem její hlavní diagonály jsou husté více, než podmatice mimo diagonálu. Část řešení indukovaného podmaticemi B_{ii} jsou tedy výpočetně nejnáročnější. Proto budeme vyžadovat, aby byly bloky B_{ii} rozděleny rovnoměrně mezi procesory, a sice tak, že blok B_{ii} (R_{ii}) přiřadíme procesoru s indexem i.

Výše popsaný přístup je zbytečně široký, nicméně máme možnost snížit definiční obor, na kterém chceme R^{opti} hledat. Pro jednoduchost předpokládejme, že časová náročnost výpočtu řešení indukovaného podmaticí B_{ij} je konstantní a rovna pro $\forall i, j \in \{1, \ldots, N\}$. Možností přiřazení bloků B_{ij} prvnímu procesoru je $\begin{pmatrix} N\,(N-1) \\ N-1 \end{pmatrix}$, druhému procesoru pak $\begin{pmatrix} (N-1)\,(N-1) \\ N-1 \end{pmatrix}$, etc.. Počet všech rovnoměrných přiřazení R pak má následující předpis:

$$\begin{aligned} \prod_{i=0}^{N-1} \begin{pmatrix} (N-1)\,(N-i) \\ N-1 \end{pmatrix} &= \frac{[N(N-1)]!}{(N-1)!\cdot[(N-1)(N-1)]!} \cdot \frac{[(N-1)(N-1)]!}{(N-1)!\cdot[(N-2)(N-1)]!} \quad \cdots \\ \cdots \quad \frac{[2(N-1)]!}{(N-1)!\cdot(N-1)!} &\cdot \frac{(N-1)!}{(N-1)!} \qquad = \\ = \frac{[N(N-1)]!}{[(N-1)!]^N} & \end{aligned}$$

což vede k asymptotické náročnosti nalezení R^{opti}: $\mathcal{O}\left(\frac{(N^2)!}{(N!)^N}\right)$.

V tabulce 1 jsou uvedeny počty přiřazení R v závislosti na N a zvoleném způsobu jejich hledání.

N	1	2	3	4	5
$\frac{N^{N^2}}{N!}$	1	4	729	$1.6777216 \cdot 10^7$	$9.53674 \cdot 10^{13}$
$\frac{(N^2)!}{(N!)^N}$	1	2	90	$3.696 \cdot 10^5$	$3.0554 \cdot 10^{11}$

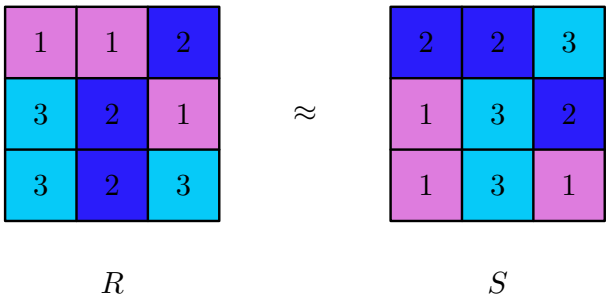
Tabulka 1: Počty přiřazení R v závislosti na N a zvoleném způsobu jejich generování. Je patrné, že řešení hrubou silou je v praxi nerealizovatelné pro $N \geq 5$.

Pro algoritmizaci hledání R^{opti} bude efektivnější, popřípadě snadnější, nahlížet na R jako na vektor o N^2 prvcích, definujme si proto linearizaci indexování R, a sice $R_k := R_{ij}$, kde $k \in \{1, \ldots, N^2\}$, $i, j \in \{1, \ldots, N\}$, $k = (i-1)\,N + j$, $i = \lceil \frac{k}{N} \rceil$ a $j = k \bmod N + 1$.

Všimněme si, že co se týče kvality řešení nezáleží na tom, zdali skupinu prvků $a \subseteq R$ přiřadíme procesoru s indexem i, skupinu prvků $b \subseteq R$ procesoru s indexem j, nebo naopak. Definujme tedy pojem izomorfismu dvou přiřazení a ukažme si, že jsme schopni hrubou silou nalézt přiřazení, která nejsou vzájemně izomorfní.

Definice 4.3 *Dvě přiřazení R a S budeme nazývat izomorfními (značit $R \approx S$) právě tehdy, když bude existovat vzájemně jednoznačné zobrazení $I : \{1, \ldots, N\} \to \{1, \ldots, N\}$ takové, že pro $\forall k \in \{1, \ldots, N^2\}$: $R_k = I\,(S_k)$.*

Pro lepší představu jsou na obrázku 8 znázorněna dvě izomorfní přiřazení.



Obrázek 8: Příklad dvou izomorfních přiřazení R a S.

Věta 4.1 *Pro libovolnou dvojici rovnoměrných přiřazení R a S platí, že $R \approx S \Rightarrow R = S$.*

Důkaz.

Důkaz povedeme přímo. Předpokládáme $R \approx S$, navíc si z definice 4.2 připomeňme, že R i S mají tu vlastnost, že $\forall i \in \{1, \ldots, N\} : R_{ii} = S_{ii} = i$, neboli $R_m = S_m = i$ (kde $m = (i-1)\,N + i$). Po začlenění definice izomorfismu I musí platit, že $\forall i \in \{1, \ldots, N\} : R_m = S_m \wedge R_m = I\,(S_m)$, z čehož vyplývá, že I musí býti identické zobrazení, a proto $R = S$.

■

Vidíme tedy, že se můžeme zaměřit na hledání rovnoměrných přiřazení a nemusíme mít obavy o generování, v jistém smyslu, redundantních řešení. Ukázka algoritmu (Algoritmus 3), který hledá R^{opti} hrubou silou s asymptotickou náročností $\mathcal{O}\left(\frac{(N^2)!}{(N!)^N}\right)$ je uvedena níže.

Algoritmus 3 rekurzivně generuje všechna rovnoměrná přiřazení tak, že vyzkouší všechna přiřazení N prvků procesoru s indexem 1, poté N prvků procesoru s indexem 2 etc.. Po přiřazení posledních prvků procesoru s indexem N je porovnána kvalita takto vygenerovaného řešení s prozatím nejlepším nalezeným řešením (ve smyslu dříve definované cenové funkce c) a zapamatuje si lepší z nich.

V inicializačním kroku algoritmu definujeme R, ze kterého budeme vycházet dle definice 4.2, tzn. že procesorům budeme přiřazovat $N-1$ prvků, jelikož přiřazení prvků na diagonále již je předem dáno.

Jádrem algoritmu 3 je rekurzivní funkce Alg1 se třemi vstupními parametry:

- $l \in \{1,..,N\}$ označuje index procesoru, kterému se algoritmus právě snaží přiřadit některou N-tici prvků R.
- $i \in \left\{1,..,N^2\right\}$ označující index prvku R, který zkusíme přiřadit procesoru s indexem l.
- $p \in \{1,..,N\}$označuje počet prvků R, které jsou právě přiřazeny procesoru s indexem l.

První podmínka ve funkci Alg1 zkoumá, zdali má smysl dále pokračovat v generování R (kontroluje, zdali je možné vygenerovat rovnoměrné R). V případě, že jsme procesoru přiřadili N prvků druhá podmínka zjišťuje, který ze dvou možných stavů nastal. Buď $l = N$, což znamená, že jsme vygenerovali rovnoměrné řešení a můžeme tak srovnat jeho kvalitu s prozatím nejlepším nalezeným řešením, nebo $l < N$ a je třeba přiřadit $N-1$ prvků dalšímu procesoru v řadě.

V případě, že $p < N$ je třeba přiřadit zbývajících $N-p$ prvků procesoru s indexem l. V případě, že pozice R_i není již přiřazena žádnému procesoru, tak algoritmus provede binární rozvoj hledání

řešení, a sice tak, že zkusí nalézt řešení kde $R_i = l$, anebo $R_i \neq l$($R_i = 0$ značí, že chceme zkusit prvek R_i přiřadit procesoru s indexem $j > l$).

Vstupy : $N \in \mathbb{N}$
Výstupy : $R^{opti} \in \mathbb{N}^{N^2}$

Inicializace: $R = \{0\}^{N^2}, \forall i \in \{1, \ldots, N\} : R_{(i-1)N+i} = i$, $R^{opti} = \{0\}^{N^2}$

funkce `Alg1` (l, i, p) **begin**
- **if** $i + N - p > N^2$ **then**
  - **return**
- **end**
- **if** $p = N$ **then**
  - **if** $l = N$ **then**
    - **if** $\left(c\left(R^{opti}\right) > c\left(R\right)\right) \vee \left(R^{opti} = \{0\}^{N^2}\right)$ **then**
      - $R^{opti} \leftarrow R$
    - **end**
  - **end**
  - **else**
    - zavoláme `Alg1` $(l + 1, 1, 1)$
  - **end**
- **end**
- **else**
  - **if** $R_i = 0$ **then**
    - $R_i \leftarrow l$
    - zavoláme `Alg1` $(l, i + 1, p + 1)$
    - $R_i \leftarrow 0$
    - zavoláme `Alg1` $(l, i + 1, p)$
  - **end**
  - **else**
    - zavoláme `Alg1` $(l, i + 1, p)$
  - **end**
- **end**

end

begin
- zavoláme `Alg1` $(1, 1, 1)$
- **return** R^{opti}

end

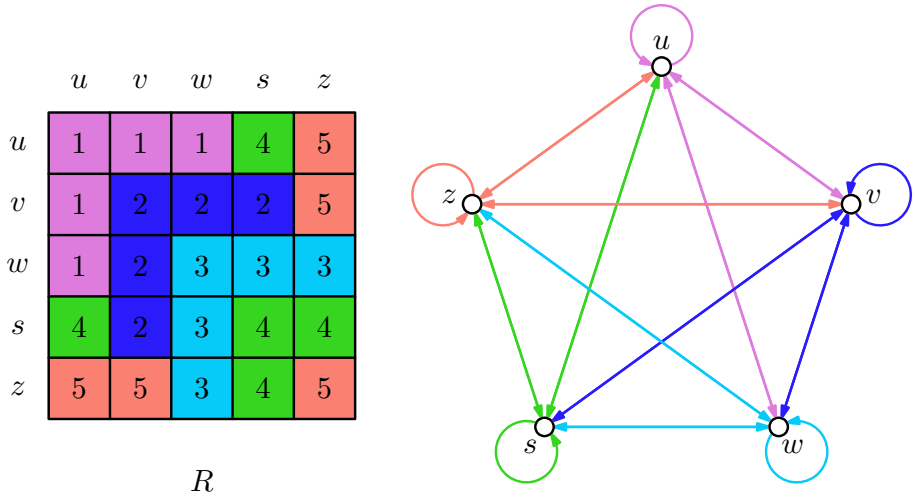
Algoritmus 3: Nalezení R^{opti} hrubou silou, bez izomorfismů.

4.3 Cyklická řešení

V této části kapitoly přeformulujeme problém hledání R^{opti} a představíme metody, které jsou schopny R^{opti} nalézt, či aproximovat. Nápadů, jak definiční obor hledání R^{opti} dále omezit je dozajista více, nicméně zde se budeme zabývat úvahou, ke které nám dopomohla teorie grafů. Základní myšlenkou bude nalezení přiřazení bloků pouze jednomu procesoru, které bude posléze použito pro odvození přiřazení zbývajících bloků ostatním procesorům, přičemž všechny procesory budou k realizaci výpočtu vyžadovat totožnou velikost paměti. Tento přístup má nejméně dvě výhody, a to snížení náročnosti výpočtu a snížení paměti potřebné pro uložení takto nalezených přiřazení. Níže jsou uvedeny základní definice, které umožní přeformulování problému nalezení R^{opti}. Budeme předpokládat, že $N \geq 3$, jelikož pro $N \leq 2$ je nalezení R^{opti} triviální.

4.3.1 Odvození

Z motivačních důvodů se nejdříve podívejme na třídu řešení, které až posléze odvodíme. Na obrázku 9 můžeme nahlédnout na spojitost mezi hodnotami v matici R a odpovídajícím hranovým ohodnocením tzv. kompletního grafu. V další části textu popíšeme tuto spojitost a vysvětlíme princip, jak vygenerovat přiřazení práce procesorům na základě hranových ohodnocení kompletních grafů.



Obrázek 9: Ukázka cyklického řešení R pro $N = 5$ společně s opovídajícím hranovým ohodnocením kompletního grafu na pěti vrcholech.

Definujme si základní pojmy, orientovaný graf a kompletní orientovaný graf a ukažme si jejich vztah k dříve definovanému řešení v podobě přiřazení R.

Definice 4.4 *Orientovaný graf G je dvojice množin (V, E), kde V je neprázdná množina vrcholů a kde E je uspořádáná množina dvojic vrcholů, zkráceně zapisujeme jako $G = (V, E)$. Buď $u, v \in V$, pak zápisem $(u, v) \in E$ budeme rozumět hranu z vrcholu u do vrcholu v, a zápisem $(v, u) \in E$ budeme rozumět hranu z vrcholu v do vrcholu u. Existenci násobných hran, tj. 2 a více hran (u, v), nebudeme uvažovat. Rovněž budeme uvažovat existenci hran (u, u) pro $\forall u \in V$. Množinu hran grafu G budeme značit jako $E(G)$, množinu vrcholů grafu G pak jako $V(G)$.*

Definice 4.5 *Kompletní orientovaný graf N vrcholech budeme značit K_N, přičemž orientovaný graf K_N má tu vlastnost, že obsahuje hrany mezi všemi uspořádanými dvojicemi vrcholů. U některých autorů se většinou setkáváme s definicemi kompletních grafů neobsahující smyčky (hran (u, u)), nicméně musím zdůraznit, že v tomto textu existenci smyček budeme vyžadovat.*

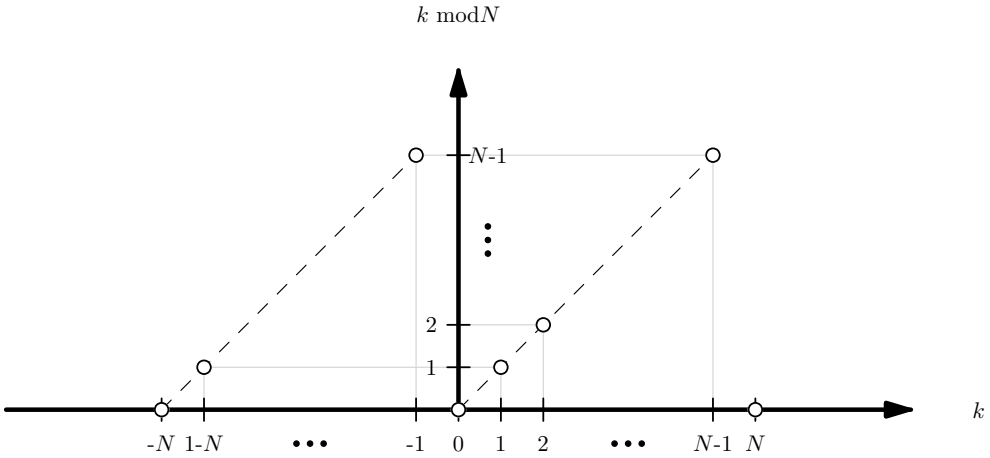
Poznámka 4.1 Počet orientovaných hran v námi definovaném grafu K_N je tedy N^2.

Definice 4.6 *Mějme graf G. Matice sousednosti $S \in \mathbb{N}^{N \times N}$, kde $N = |V(G)|$, vyjadřuje vztah mezi všemi dvojicemi vrcholů grafu G. Definujme si prosté zobrazení $t : V(G) \to \{0, 1, \ldots, N-1\}$, které v další části textu budeme nazývat **ohodnocením vrcholů** grafu G. Pro $\forall (u, v) \in E(G)$ bude $S_{ij} = 0$ pokud $(u, v) \notin E(G)$ a $S_{ij} = 1$ pokud $(u, v) \in E(G)$, kde $i = t(u)$ a $j = t(v)$.*

Poznámka 4.2 Všimněme si, že libovolné přiřazení R a matice sousednosti S mají totožnou dimenzi, můžeme tedy na libovolný prvek matice R nahlížet jako na hranu grafu K_N.

Všímaví čtenáři možná v motivačním obrázku 9 rozpoznali určitý vzor, a sice, že hrany obarvené jednou barvou byly v určitém smyslu rotovány ve směru hodinových ručiček. Pro formální definici operace rotace hran grafu K_N a pro dokázání její jednoznačnosti zavedeme dva pojmy, míru orientace hrany a délku hrany.

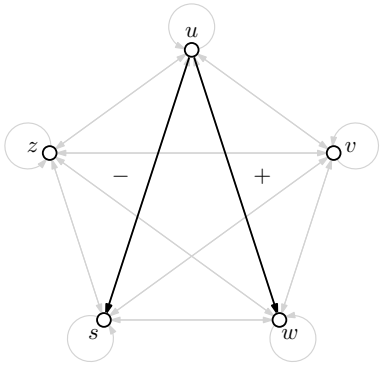
Pro připomenutí zmíníme vlastnost operace modulo, kterou budeme dále využívat. Mějme $a \in \mathbb{Z}$, pak operaci a modulo N značíme $a \bmod N = a - kN$, kde $k \in \mathbb{Z}$ je takové, že $a - kN = \min_{\forall l \in \mathbb{Z}: a - lN \geq 0} \{a - lN\}$. Na obrázku 10 je graficky znázorněn vliv operace modulo N na přirozená čísla v rozsahu $\langle -N,\ N \rangle$.



Obrázek 10: Grafické znázornění operace modulo N.

Definice 4.7 *Mějme orientovaný graf K_N, libovolnou orientovanou hranu $e = (u, v) \in E(K_N)$, vrcholové ohodnocení t grafu G a zobrazení $d : E(K_N) \to \{0, 1, \ldots, N-1\}$, jež budeme v další části textu nazývat* ***mírou orientace hrany*** *e, kde $d(e) = [t(v) - t(u)]\, mod N$. O hraně e řekneme, že je:*

- ***Kladně orientována**, právě tehdy, když $d(e) < \frac{N}{2}$.*
- ***Záporně orientována**, právě tehdy, když $d(e) > \frac{N}{2}$.*
- *Pokud $d(e) = 0 \vee d(e) = \frac{N}{2}$, pak orientaci hrany e nebudeme uvažovat.*



Obrázek 11: Ukázka dvou různých typů orientovaných hran pro graf, ve kterém platí, že $t(u) = 0$, $t(v) = 1, t(w) = 2, t(s) = 3$ a $t(z) = 4$. Hrana ohodnocena $+$ je kladně orientována, hrana ohodnocena $-$ je záporně orientována.

Definice 4.8 *Mějme kompletní graf K_N, jeho vrcholové ohodnocení t a míru orientace hran d (viz definice 4.7). Délku libovolné hrany $e = (u, v) \in E(K_N)$ pak definujeme jako zobrazení $l : E(K_N) \to \mathbb{N}$, kde $l(e) = \min\{d(e),\ N - d(e)\}$.*

Věta 4.2 *Mějme libovolnou orientovanou hranu $e = (u, v) \in E(K_N)$, označme si hranu $f = (v, u)$. Platí, že $\left(d(e) < \frac{N}{2}\right) \Leftrightarrow \left(d(f) > \frac{N}{2}\right)$ a navíc, že $l(e) = l(f)$.*

Důkaz.

Mějme hranu $e = (u, v)$, pak míra její orientace je $d(e) = (t(v) - t(u)) \bmod N$ a $\exists k \in \mathbb{Z}$ takové, že $d(e) = t(v) - t(u) - kN$. Přesněji,

- $t(v) > t(u) \Rightarrow d(e) = t(v) - t(u)$, jelikož $f = (v, u)$, pak $d(f) = t(u) - t(v) + N = N - d(e)$.
- $t(v) < t(u) \Rightarrow d(e) = t(v) - t(u) + N$, jelikož $f = (v, u)$, pak $d(f) = t(u) - t(v) = N - d(e)$.
- Pokud $d(e) < \frac{N}{2}$, pak platí, že $N - d(e) = d(f) > \frac{N}{2}$, pokud naopak $d(e) > \frac{N}{2}$, pak $N - d(e) = d(f) < \frac{N}{2}$ a tvrzení $\left(d(e) < \frac{N}{2}\right) \Leftrightarrow \left(d(f) > \frac{N}{2}\right)$ platí.

Délka hrany e je definována jako $l(e) = \min\{d(e),\ N - d(e)\}$. Délka hrany f tudíž bude $l(f) = \min\{d(f),\ N - d(f)\} = \min\{N - d(e),\ N - (N - d(e))\} = l(e)$, a tvrzení $l(e) = l(f)$ je rovněž dokázáno.

■

V záverečné části této kapitoly nás budou zajímat počty orientovaných hran jednotlivých délek v závislosti na sudosti, či lichosti N. Ukažme si tedy závislost počtu hran všech délek grafu K_N na N a jeho paritě.

Věta 4.3 *Mějme kompletní graf K_N, míru orientace hran d a délku hran l, pak:*

- *Orientovaných hran $e \in E(K_N)$ takových, že $l(e) = 0$ je právě N (pouze smyčky).*
- *Pokud je N liché, pak pro $\forall k \in \left\{1, \ldots, \frac{N-1}{2}\right\}$ graf K_N obsahuje N kladně orientovaných hran délek k a N záporně orientovaných hran délek k.*
- *Pokud je N sudé, pak pro $\forall k \in \left\{1, \ldots, \frac{N}{2} - 1\right\}$ graf K_N obsahuje N kladně orientovaných hran délek k, N záporně orientovaných hran délek k a N hran o délce $\frac{N}{2}$.*

Důkaz.

- Zvolme si libovolný vrchol $u \in V(K_N)$. Mějme vrcholové ohodnocení t grafu K_N a definujme množinu $D = \{(t(v) - t(u)) \bmod N : v \in V(K_N)\}$. Pak $D = \{0, 1, \ldots, N-1\}$, neboť $t(v) \in \{0, 1, \ldots, N-1\}$.
- Buď N **liché**. Množina orientovaných hran $M_u = \{e = (u, v) : v \in V(K_N)\}$ obsahuje právě jednu hranu nulové délky ((u, u)), $\frac{N-1}{2}$ kladně orientovaných hran ($d(e) < \frac{N}{2}$) a $\frac{N-1}{2}$ záporně orientovaných hran ($d(e) > \frac{N}{2}$). Délky kladně orientovaných hran e tvoří množinu $\left\{1, \ldots, \frac{N-1}{2}\right\}$ ($l(e) = d(e)$) a délky záporně orientovaných hran rovněž tvoří množinu $\left\{1, \ldots, \frac{N-1}{2}\right\}$ ($l(e) = N - d(e)$).
- Buď N **sudé**. Množina orientovaných hran $M_u = \{e = (u, v) : v \in V(K_N)\}$ obsahuje právě jednu hranu nulové délky ((u, u)), právě jednu hranu délky $\frac{N}{2}$, $\frac{N}{2} - 1$ kladně orientovaných hran ($d(e) < \frac{N}{2}$) a $\frac{N}{2} - 1$ záporně orientovaných hran ($d(e) > \frac{N}{2}$). Délky kladně orientovaných hran e tvoří množinu $\left\{1, \ldots, \frac{N}{2} - 1\right\}$ ($l(e) = d(e)$) i délky záporně orientovaných hran tvoří množinu $\left\{1, \ldots, \frac{N}{2} - 1\right\}$ ($l(e) = N - d(e)$).
- Odvozené vztahy platí pro $\forall u \in V(K_N)$ současně, jelikož pro $\forall u, v \in V(K_N)$, $u \neq v$: $M_u \cap M_v = \emptyset$. A tedy:
  - pokud je N liché, tak K_N obsahuje $N\left(\frac{N-1}{2}\right)$ záporně orientovaných hran, z toho N hran délky 1, N hran délky 2,..., N hran délky $\frac{N-1}{2}$, $N\left(\frac{N-1}{2}\right)$ kladně orientovaných hran, z toho N hran délky 1, N hran délky 2,..., N hran délky $\frac{N-1}{2}$ a N hran nulové délky.
  - pokud je N sudé, tak K_N obsahuje $N\left(\frac{N}{2} - 1\right)$ záporně orientovaných hran, z toho N hran délky 1, N hran délky 2,..., N hran délky $\frac{N}{2} - 1$, $N\left(\frac{N}{2} - 1\right)$ kladně orientovaných hran, z toho N hran délky 1, N hran délky 2,..., N hran délky $\frac{N}{2} - 1$, N hran nulové délky a N hran o délce $\frac{N}{2}$.

■

Základním kamenem formulace cyklických řešení je rotace hrany, popřípadě množin hran.

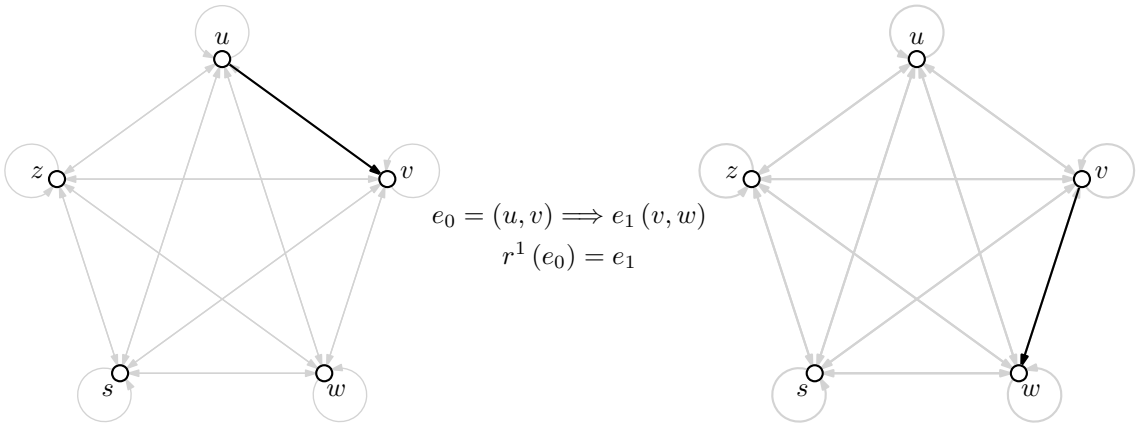
Definice 4.9 *Mějme graf K_N a libovolnou hranu $e_0 = (u_0, v_0) \in E(K_N)$. Rotaci hrany pak definujeme jako zobrazení $r : E(K_N) \to E(K_N)$ takové, že $r(e_0) = e_1 = (u_1, v_1)$, kde $t(u_1) = (t(u_0) + 1) \bmod N$ a $t(v_1) = (t(v_0) + 1) \bmod N$.*

j-násobnou rotaci hrany e_0 označíme jako $r^j(e_0)$, kde $r^j(e_0) = (u_j, v_j) := r \circ r \circ \cdots \circ r(e_0)$ je taková hrana, pro kterou platí, že $t(u_j) = (t(u_0) + j) \bmod N$ a $t(v_j) = (t(v_0) + j) \bmod N$.

j-násobnou rotací množiny hran $H \subseteq E(K_N)$ budeme rozumět j-násobnou rotaci všech hran $e \in H$.

Poznámka 4.3 Platnost předpisu pro j-násobnou rotaci můžeme odvodit na základě postupné jednotkové rotace.

1. Mějme hranu $e_0 = (u_0, v_0) \in E(K_N)$, pak $r(e_0) = e_1 = (u_1, v_1)$ je taková hrana, pro kterou platí, že $t(u_1) = (t(u_0) + 1) \bmod N$ a $t(v_1) = (t(v_0) + 1) \bmod N$.
2. Předpokládejme existenci j-krát rotované hrany e_0, čili $e_j = r^j(e_0) = (u_j, v_j)$, kde $t(u_j) = (t(u_0) + j) \bmod N$ a $t(v_j) = (t(v_0) + j) \bmod N$.
3. Pak rotací $r(e_j) = e_{j+1} = (u_{j+1}, v_{j+1})$, pro kterou platí, že $t(u_{j+1}) = (t(u_j) + 1) \bmod N$ a $t(v_{j+1}) = (t(v_j) + 1) \bmod N$.
4. Po dosazení hodnot $t(u_j)$ a $t(v_j)$ z bodu 2 do výrazů v bodě 3 zjistíme, že:
   - $t(u_{j+1}) = ((t(u_0) + j) \bmod N + 1) \bmod N = ((t(u_0) + j) - kN + 1) \bmod N = (t(u_0) + (j + 1)) \bmod N$.
   - Analogicky odvodíme předpos pro hodnotu $t(v_{j+1})$.



Obrázek 12: Ukázka rotace jedné hrany kde $j = 1$, $t(u) = 0$, $t(v) = 1$, $t(w) = 2$, $t(s) = 3$ a $t(z) = 4$.

Věta 4.4 *Buď $e_0 = (u_0, v_0) \in V(K_N)$ a $j \in \{1, \ldots, N-1\}$: libovolné. Platí, že $r^j(e_0) \neq e_0$ a $r^N(e_0) = e_0$.*

Důkaz.

$r^N(e_0) = e_0$:

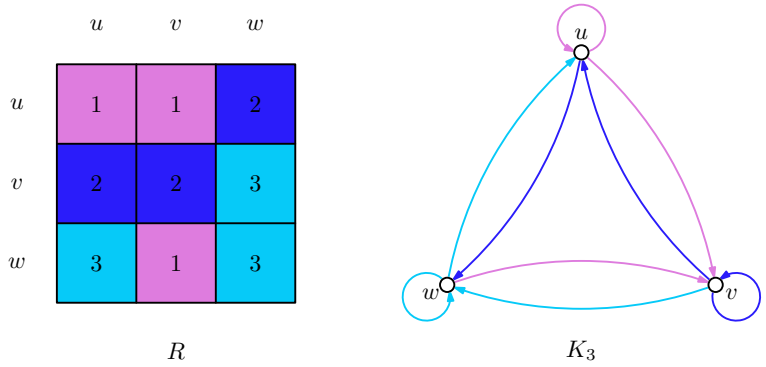
- $r^N(e_0) = e_N = (u_N, v_N)$ je taková hrana, pro kterou platí, že $t(u_N) = (t(u_0) + N) \bmod N = t(u_0)$ a $t(v_N) = (t(v_0) + N) \bmod N = t(v_0)$, a proto $e_N = e_0$.

$\forall j \in \{1, \ldots, N-1\} : r^j(e_0) \neq e_0$:

- Označme $r^j(e_0) = e_j = (u_j, v_j)$, přičemž $t(u_j) = (t(u_0) + j) \bmod N$. Z vlastností operace modulo N víme, že $t(u_j) \neq t(u_0)$, ledaže by $j = kN$ pro $\forall k \in \mathbb{Z}$. Víme tedy, že libovolná hrana
- označme P jako množinu vrcholových ohodnocení počátečních vrcholů hran $r^j(e_0)$, tedy $P = \{t(u_j) : j \in \{1, \ldots, N-1\}, e_j = (u_j, v_j) = r^j(e_0)\}$. P obsahuje $N-1$ prvků, neboť $t(u_j) = (t(u_0) + j) \bmod N$. Jelikož $|P| = N - 1$, pak $r^j(e_0)$

■

Nyní již jsme schopni problém nalezení R^{opti} formulovat za využití teorie grafů. Mějme prvek matice R na i-tém řádku a v j-tém sloupci. Pak na R_{ij}můžeme nahlížet jako na orientovanou hranu (u, v) v grafu K_N takovou, že $t(u) = i$ a $t(v) = j$. Pro výpočet bloku R_{ij} budeme potřebovat datové bloky D_i a D_j, tudíž můžeme na vrcholy grafu K_N nahlížet jako na reprezentanty právě těchto datových bloků. Naším úkolem je ohodnotit hrany grafu K_N (například očíslováním hran indexy procesorů) tak, aby byly hrany náležící jednomu procesoru incidentní s minimálním počtem vrcholů.



Obrázek 13: Vztah mezi přiřazením R a ohodnocením kompletního grafu K_3.

Přednesme si důležité tvrzení, které využijeme v pozdějším důkazu správnosti formulace cyklických řešení.

Věta 4.5 *Mějme libovolnou hranu $e \in E(K_N)$. Platí, že míra orientace hrany $d(e)$ a délka hrany $l(e)$ jsou invariantní vůči rotaci hrany $r(e)$.*

Důkaz.

$d(e) = d(r(e))$:

- Označme si $e = (u_0, v_0)$.
- Mějme hranu $f = (u_1, v_1) = r(e)$. Pak $d(f) = [t(v_1) - t(u_1)] \bmod N$, $t(u_1) = (t(u_1) + 1) \bmod N$ a $t(v_1) = (t(v_0) + 1) \bmod N$, a proto tedy $d(f) = [(t(v_0) + 1) \bmod N - (t(u_0) + 1) \bmod N] \bmod N$.
- Buď $a \in \mathbb{Z}$ takové, že $(t(u_0) + 1) \bmod N = t(u_0) + 1 - aN$, rovněž buď $b \in \mathbb{Z}$ takové, že $(t(v_0) + 1) \bmod N = t(v_0) + 1 - bN$.
- Pak $d(f) = [(t(u_0) + 1 - aN) - (t(v_0) + 1 - bN)] \bmod N = [t(u_0) + 1 - t(v_0) - 1] \bmod N = [t(u_0) - t(v_0)] \bmod N = d(e)$.

$l(e) = l(f)$:

- Víme, že $l(e) = \min\{|d(e)|, N - |d(e)|\}$ a $l(f) = \min\{|d(f)|, N - |d(f)|\}$. Rovněž jsme ukázali, že $d(e) = d(f)$, a proto $l(e) = l(f)$.

■

Nyní již jsme schopni dokázat tvrzení, které nám umožní formálně přeformulovat problém přiřazení práce procesorům za využití teorie grafů.

Věta 4.6 *Definujme množinu $H_0 \subseteq E(K_N)$ čítající N prvků a obsahující právě jednu kladně orientovanou hranu každé délky, právě jednu záporně orientovanou hranu každé délky, jednu hranu nulové délky a v případě, že je N sudé i jednu orientovanou hranu délky $\frac{N}{2}$. j-násobnou rotaci množiny H_0 označme H_j, kde $j \in \{1, 2, \ldots, N-1\}$. Pokud ohodnotíme hrany grafu K_N indukované množinami H_j hodnotou $j + 1$ pro $\forall j \in \{0, 1, , N-1\}$, pak budou ohodnoceny všechny hrany grafu K_N, přičemž pro $\forall l \in \{1, \ldots, N\}$ bude platit, že orientovaných hran ohodnocených hodnotou l bude právě N.*

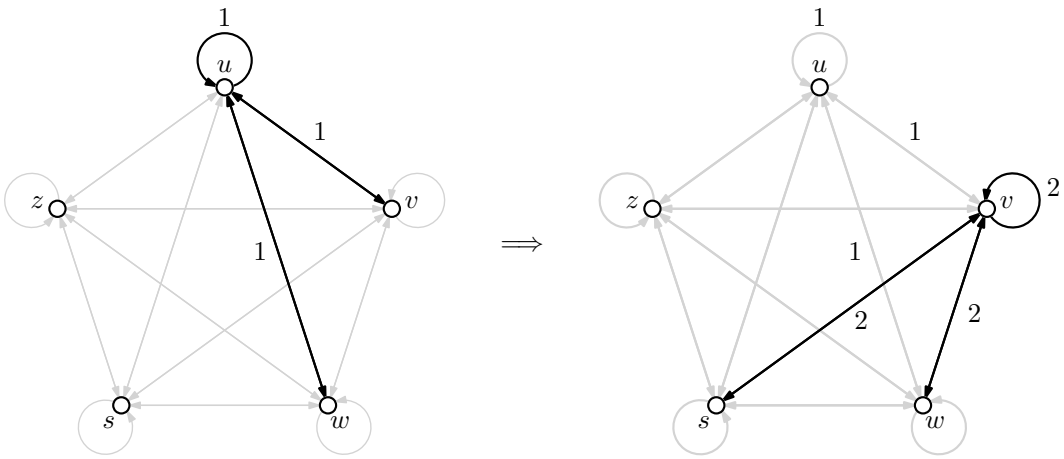
Důkaz.

- Zvolme libovolnou množinu hran $H_0 \subseteq E(K_N)$ takovou, aby splňovala předpoklady věty 4.6. Dle definice rotace množin hran dále definujme množiny $H_1, H_2, \ldots, H_{N-1}$, kde $H_i = \{r(e) : e \in H_{i-1}\}$ pro $\forall i \in \{1, 2, \ldots, N-1\}$, tzn., že množinu hran H_i obdržíme rotací hran náležících množine H_{i-1}. Když hranám náležících množine H_i přiřadíme hodnotu i, pro $\forall i \in \{0, 1, \ldots, N-1\}$, tak obdržíme ohodnocení grafu K_N. Je však třeba ukázat, že takovéto ohodnocení je správné, čili, že je ohodnocena každá orientovaná hrana grafu K_N, a že počet hran ohodnocených hodnotou i je právě N.

- Pro $\forall i, j \in \{0, 1, \ldots, N-1\}$, kde $i \neq j$ je třeba ukázat, že $H_i \cap H_j = \emptyset$(průnik libovolné rozdílné dvojice množin hran je prázdný, neboli, že hrany dvou různých H_i a H_j se nijak nepřekrývají). Platnost tohoto tvrzení by implikovala, že počet hran ohodnocených i je roven N pro $\forall i \in \{0, 1, \ldots, N-1\}$, jelikož $|H_i| = N$, $\forall i \in \{0, 1, \ldots, N-1\}$, rovněž by platilo, že by byly ohodnoceny všechny hrany grafu K_N, protože počet různých množin H_i je N, každá s N hranami a tudíž $\left|\bigcup_{i=0}^{N-1} H_i\right| = \sum |H_i| = N^2$, což odpovídá počtu orientovaných hran v grafu K_N.

- Z věty 4.5 víme, že délky a míry orientace hran jsou invariantní vůči rotaci, a proto při dokazování pravdivosti tvrzení $\forall i, j \in \{0, 1, \ldots, N-1\}, i \neq j : H_i \cap H_j = \emptyset$ stačí ukázat, že pro libovolnou hranu $e \in E(K_N)$ platí, že $\forall k \in \{1, \ldots, N-1\} : r^k(e) \neq e$, o čemž víme, že je pravda z věty 4.4.

■



Obrázek 14: Ukázka rotace množiny hran a jejího inkrementálního ohodnocení pro graf, kde $t(u) = 0$, $t(v) = 1$, $t(w) = 2$, $t(s) = 3$ a $t(z) = 4$.

Důsledky: Z věty 4.6 tedy vyplývá, že můžeme použít **libovolnou** množinu hran H splňující předpoklady věty a použít ji pro ohodnocení všech hran grafu K_N tak, aby bylo právě N hran ohodnocených hodnotou 1, N hran ohodnocených hodnotou 2 etc... V důsledku je tedy postačující zaměřit se na nalezení takové množiny H, že graf indukovaný množinou hran H vyžaduje co nejmenší počet vrcholů (vrcholy reprezentují datové bloky pro realizaci výpočtu hraniční úlohy).

Hranové ohodnocení grafu K_N vyplývající z věty 4.6 rovněž bude splňovat požadavek uvedený v definici 4.2, jelikož H_i obsahuje právě jednu hranu nulové délky (v matici sousednosti S je tato hrana na pozici S_{ii}) a v důsledku je tedy každý diagonální blok přiřazen jinému procesoru.

4.3.2 Algoritmizace

V této části si popíšeme způsoby nalezení množiny H splňující předpoklady věty 4.6 a vyžadující nejmenší počet vrcholů (označme takovou množinu jako H^{opti}), přičemž odvození odpovídajícího přiřazení R vyplývá definice rotace (definice 4.9). Nejdříve popíšeme způsob nalezení hrubou sílou, vyplývající přímo z věty 4.6, poté se zaměříme na optimalizaci nalezení H^{opti} s využitím znalosti minimalizačního charakteru problému a představíme deterministické i nedetreministické přístupy, jak H^{opti} nalézt, či aproximovat.

Triviální algoritmy Nalezení H^{opti} můžeme zrealizovat provedením všech N prvkových výběrů z množiny orientovaných hran $E(K_N)$, přičemž z nich vybereme ten, který vyžaduje nejmenší počet různých vrcholů. Řádová složitost tohoto přístupu je $\mathcal{O}\left(\left(\begin{array}{c} N^2 \\ N \end{array}\right)\right) = \mathcal{O}\left(\frac{(N^2)!}{N!\cdot(N^2-N)!}\right)$.

Uvědomme si však, že o některých N prvkových výběrech z $E(K_N)$ víme, že nesplňují předpoklady věty 4.6, zaměřme se proto na generování takových výběrů, které tyto předpoklady splňují. Rozdělme si $E(K_N)$ podle délek a orientací hran na vzájemně disjunktní množiny, a sice $E(K_N) = \bigcup_{i=1}^{\lfloor \frac{N}{2} \rfloor} \left(M_i^+ \cup M_i^-\right) \cup M_0 \cup M_{\frac{N}{2}}$, kde M_i^+ značí množinu kladně orientovaných hran délky i, M_i^- značí množinu záporně orientovaných hran délky i, M_0 značí množinu hran délky 0 a $M_{\frac{N}{2}}$ značí množinu hran délky $\frac{N}{2}$. Na základě této znalosti můžeme libovolnou množinu H sestrojit tak, že vybereme právě jeden prvek z množin M_i^+, $\forall i : 1 \leq i < \frac{N}{2}$, jeden prvek z množiny M_i^-, $\forall i : 1 \leq i < \frac{N}{2}$, jeden prvek z M_0 a jeden prvek z $M_{\frac{1}{2}}$.

Asymptotická složitost tohoto přístupu je $\mathcal{O}\left(N^{\frac{N}{2}} \cdot N^{\frac{N}{2}} \cdot N \cdot N\right) = \mathcal{O}\left(N^N\right)$. Nicméně máme možnost nalezení H^{opti} dále urychlit díky vlastnosti hrany $e = (u, v) \in E(K_N)$ takové, že: $(u, v) \in M_i^+ \Leftrightarrow (v, u) \in M_i^-$(viz věta 4.2), což v důsledku znamená, že je postačující vybrat libovolnou hranu z M_i^+, čímž přímo definujeme hranu, kterou můžeme zvolit z M_i^- a zařadit ji do cyklického řešení aniž bychom navýšili počet použitých vrcholů.

Dále do algoritmu nemusíme zařazovat hledání hrany nulové délky, jelikož tuto hranu můžeme zařadit do libovolného řešení bez navýšení použitých vrcholů. Reálná složitost tohoto přístupu je $N^{\frac{N-1}{2}-1}$ pro lichá N, popřípadě $N^{\frac{N}{2}-1}$ pro N sudá, asymptotická složitost je pak $\mathcal{O}\left(\sqrt{N}^N\right)$. Algoritmus produkující H^{opti} na základě úvah uvedených výše je k nahlédnutí níže.

Algoritmus 4 generuje všechny kombinace hran splňující předpoklady věty 4.6 (viz popis výstupu v algoritmu). Jádrem algoritmu je rekurzivní funkce Alg2(i), která do globálně uloženého řešení H postupně zkusí zařadit každou hranu délky i společně s následným voláním funkce Alg2$(i+1)$ pro

vyzkoušení zařazení všech hran délek $i+1$. Ukončovací podmínkou algoritmu je stav, kdy se do H snažíme zařadit hranu délky $i > \frac{N}{2}$, přičemž v této situaci si zapamatujeme prozatím nejlepší nalezené řešení. Na závěr algoritmus doplní do H^{opti} hranu délky 0 a poté vrátí výsledek.

Vstupy : $N \in \mathbb{N}$

Výstupy : H^{opti}, množina orientovaných hran obsahující jednu hranu nulové délky, pro $i \in \mathbb{N}$ taková, že $1 \leq i < \frac{N}{2}$ H obsahuje právě jednu hranu délky i kladné i záporné orientace a pokud je N sudé, tak i jednu libovolnou orientovanou hranu délky $\frac{N}{2}$ vyžadující nejmenší počet vrcholů

Inicializace: $\forall i \in \left\{0, \ldots, \left\lceil \frac{N}{2} - 1 \right\rceil\right\}$, $M_i = \{e = (u,v) \in E(K_N) : t(u) \leq t(v) \wedge l(e) = i\}$, $M_{\frac{N}{2}} = \left\{e \in E(K_N) : l(e) = \frac{N}{2}\right\}$, $H^{opti} = \emptyset$, $H = \emptyset$

funkce `Alg2(i)` **begin**

- **if** $i > \frac{N}{2}$ **then**
  - **if** $H^{opti} = \emptyset$ **then**
    - $H^{opti} \leftarrow H$
  - **end**
  - **else if** $\left|\left\{u \in V(K_N) : u \subseteq e \in H^{opti}\right\}\right| > \left|\left\{u \in V(K_N) : u \subseteq e \in H\right\}\right|$ **then**
    - $H^{opti} \leftarrow H$
  - **end**
  - **return**
- **end**
- **for** $\forall e \in M_i$ **do**
  - $H \leftarrow H \cup \{e\}$
  - zavoláme `Alg2(i+1)`
  - $H \leftarrow H \setminus \{e\}$
- **end**

end

begin

- zavoláme `Alg2(1)`
- $H^{opti} \leftarrow H_{opti} \cup \{e \in \{(u,u) : u \subset f \in H\}\}$
- **return** H^{opti}

end

Algoritmus 4: Nalezení H^{opti} triviálním způsobem.

Heuristické algoritmy Algoritmus 4 je opět časově velice náročný a na běžných počítačích nerealizovatelný zhruba pro $N \geq 20$. Máme však možnost využít znalosti minimalizačního požadavku na počet vrcholů již v průběhu generování jednoho řešení, a ne jen při kontrole kvality řešení právě vygenerovaného. V algoritmu 4 byla definičním oborem množina hran $E(K_N)$, nicméně můžeme problém nalezení H^{opti} přeformulovat tak, aby definičním oborem byla množina vrcholů $V(K_N)$, přičemž v průběhu algoritmu budeme definiční obor dále deterministicky omezovat, aniž bychom zamezili možnosti H^{opti} nalézt.

Algoritmus 5 pracuje na rekurzivním sestavování množiny vrcholů M (v jednom kroku do M zařadí jeden vrchol), přičemž v každém kroku kontroluje, zdali je možno na množině vrcholů M indukovat množinu hran H splňující předpoklady věty 4.6. Algoritmus v každém kroku sestavuje množinu kandidátských vrcholů C_k na zařazení do řešení a postupně zkusí do M zařadit každý vrchol z kandidátské množiny (deterministický přístup), přičemž kandidátská množina C_k obsahuje ty vrcholy, jež by po zařazení do M indukovaly nejvyšší počet orientovaných hran chybějících délek a orientací.

Výstupem algoritmu je množina hran H. Pro vyšší N s příliš obsáhlými stromy řešení je do algoritmu zařazena omezující podmínka ve formě parametru $k \in \mathbb{N}$, jenž způsobí platnost nerovnosti $|C_k| \leq k$.

Vstupy : $N \in \mathbb{N}$, $k \in \mathbb{N}$

Výstupy : H, množina orientovaných hran obsahující jednu hranu nulové délky, pro $i \in \mathbb{N}$ taková, že $1 \leq i < \frac{N}{2}$ H obsahuje právě jednu hranu délky i kladné i záporné orientace a pokud je N sudé, tak i jednu libovolnou orientovanou hranu délky $\frac{N}{2}$

Inicializace: $\forall i \in \left\{0, \ldots, \left\lceil \frac{N}{2} - 1 \right\rceil\right\}$, $M_i = \{e = (u,v) \in E(K_N) : t(u) \leq t(v) \wedge l(e) = i\}$, $M_{\frac{N}{2}} = \left\{e \in E(K_N) : l(e) = \frac{N}{2}\right\}$, $H = \emptyset$, $M = \emptyset$, $M^{opti} = \emptyset$, $Z = \emptyset$

funkce `Alg3(`M_{in}`)`

begin

$L \leftarrow \{l(e) : u, v \in M_{in},\ e = (u,v)\}$

if $|L| = \left\lfloor \frac{N}{2} \right\rfloor + 1$ **then**

if $M^{opti} = \emptyset$ **then**

$M^{opti} \leftarrow M_{in}$

end

else if $|M_{in}| < \left|M^{opti}\right|$ **then**

$M^{opti} \leftarrow M_{in}$

end

return

end

if $\left(|M_{in}| \geq \left|M^{opti}\right| - 1\right) \wedge \left(M^{opti} \neq \emptyset\right)$ **then**

return

end

$$C \leftarrow \left\{ v \in V(K_N) : |\{l(e) : u \in M_{in},\ e = (u,v)\}| = \max_{w \in V(K_N)} |\{l(f) : u \in M_{in},\ f = (u,w)\}| \right\}$$

if $|C| \leq k$ **then**

$C_k \leftarrow C$

end

else

$C_k \subseteq C : |C_k| = k \wedge \forall v \in C_k, \forall u \in C \setminus C_k : t(u) < t(v)$

end

for $\forall u \in C_k$ **do**

$M_{in} \leftarrow M_{in} \cup \{u\}$

zavoláme `Alg3(`M_{in}`)`

$M_{in} \leftarrow M_{in} \setminus \{u\}$

end

end

begin

zavoláme `Alg3(`M`)`

for $\forall i \in \left\{0, \ldots, \frac{N}{2}\right\}$ **do**

$H \leftarrow H \cup \left\{f \in \left\{e = (u,v) \in E(K_N) : \{u,v\} \subseteq M^{opti} \wedge e \in M_i\right\}\right\}$

end

return H

end

Algoritmus 5: Nalezení aproximace H^{opti} heuristicky, s možným omezením počtu prvků kandidátských množin

Intuitivně lze možná cítit, že algoritmus 5 produkuje dobrá řešení, nicméně zbývá určit zdali je schopen nalézt řešení nejlepší. Dokažme si proto následující tvrzení.

Věta 4.7 *Algoritmus 5 bez omezení počtu prvků kandidátské množiny vrcholů C nalezne H^{opti}.*

Důkaz.

Důkaz povedeme sporem.

Buď M libovolnou množinou vrcholů indukující nejlepší nalezené řešení H algoritmem 5. Dále předpokládejme existenci množiny vrcholů $\overline{M}$ indukující kvalitnější řešení $\overline{H}$, tzn. $\left|\overline{M}\right| < |M|$. Bez újmy na obecnosti můžeme předpokládat, že $\left|\overline{M} \cap M\right| \geq 2$, jelikož $\overline{M}$ i M indukují hrany všech délek a můžeme tedy buď $\overline{H}$ nebo H orotovat tak, aby H a $\overline{H}$ měly společnou alespoň jednu hranu (dva vrcholy).

Dále si vezměme libovolnou podmnožinu vrcholů $W \subseteq \overline{M} \cap M$ takovou, že $\nexists w \in W$ takové, které by bylo algoritmem 5 zařazeno do M v i-tém pořadí, přičemž $i > |W|$ (důvod pro tento požadavek spočívá v exaktní specifikaci stavu, ve kterém se může nacházet řešení právě generované algoritmem 5). Bez újmy na obecnosti můžeme předpokládat, že $|W| \geq 2$, jelikož $\overline{M} \cap M$ mají společnou alespoň jednu hranu (u, v), přičemž u a v mohou býti prvními vrcholy zařazenými do M.

Jelikož předpokládáme, že $\left|\overline{M}\right| < |M|$, pak musí být pravda, že $\exists v \in \overline{M} \setminus M$ takové, že $\forall u \in M \setminus \overline{M}$ platí, že $W \cup \{v\}$ indukuje množinu s vyšším počtem hran různých délek, než množina $W \cup \{u\}$. Což však vede ke sporu, jelikož algoritmus 5 volí vrcholy k rozšíření řešení tak, aby přidávaly nejvyšší počet hran nových délek, a proto $\left|\overline{M}\right| = |M|$.

■

Metaheuristické algoritmy Algoritmus 5 buď prohledává celý definiční obor, nebo jej omezuje deterministicky, pro libovolné N pokaždé totožně a zamezí tak jakékoliv možnosti procházet tu část stromu řešení, která může obsahovat řešení kvalitnější. Máme však možnost uchýlit se k algoritmům metaheuristickým, neboli náhodným. V této části uvedeme řadu algoritmů založených na náhodném výběru. Představíme dva algoritmy založené na technice náhodného restartu a jeden evoluční algoritmus, který na algoritmech náhodného restartu staví a dále je rozšiřuje.

Nejdříve si definujeme algoritmus 6, jenž je založen na technice náhodného restartu, přičemž vrcholy jsou do H zařazovány zcela náhodně a bez jakékoliv znalosti struktury problému. Důvod zařazení tohoto algoritmu do textu souvisí s níže definovaným evolučním algoritmem a s faktem, že řešení vy-

generována s využitím minimalizačního charakteru úlohy mají tendenci býti velice blízko lokálnímu minimu a evoluční algoritmus tak má malou šanci takové řešení dále zlepšit.

Vstupy : $N \in \mathbb{N}$, cas, $tolerance$

Výstupy : H, množina orientovaných hran obsahující jednu hranu nulové délky, pro $i \in \mathbb{N}$ taková, že $1 \leq i < \frac{N}{2}$ H obsahuje právě jednu hranu délky i kladné i záporné orientace a pokud je N sudé, tak i jednu libovolnou orientovanou hranu délky $\frac{N}{2}$

Inicializace: $\forall i \in \left\{0, \ldots, \left\lceil \frac{N}{2} - 1 \right\rceil\right\}$, $M_i = \{e = (u, v) \in E(K_N) : t(u) \leq t(v) \wedge l(e) = i\}$, $M_{\frac{N}{2}} = \left\{e \in E(K_N) : l(e) = \frac{N}{2}\right\}$, $H = \{\emptyset\}$, $M = \{\emptyset\}$, $M^{opti} = V(K_N)$

funkce `Alg4(`M_{in}`)` **begin**
    $u \leftarrow$ **náhodný vrchol** $\in V(K_N) \setminus M_{in}$
    $M_{in} \leftarrow M_{in} \cup \{u\}$
    return `Alg4(`M_{in}`)`
end

begin
    while $\left(\left|M^{opti}\right| > tolerance \wedge (\textit{máme dostatek času})\right)$ **do**
        $M \leftarrow$ `Alg4(`$\emptyset$`)`
        if $|M| < \left|M^{opti}\right|$ **then**
            $M^{opti} \leftarrow M$
        end
    end
    for $\forall i \in \left\{0, \ldots, \frac{N}{2}\right\}$ **do**
        $H \leftarrow H \cup \left\{f \in \left\{e = (u, v) \in E(K_N) : \{u, v\} \subseteq M^{opti} \wedge e \in M_i\right\}\right\}$
    end
    return H
end

Algoritmus 6: Nalezení aproximace H^{opti} metaheuristicky, náhodným generováním.

Dále si představíme algoritmus 7, jenž je založen opět na technice náhodného restartu, a který funguje podobně jako algoritmus 5 s parametrem $k = 1$, jen s tím rozdílem, že v každém kroce vybere vrchol z kandidátské množiny náhodně. Takto vygenerované řešení porovná s prozatím nejlepším nalezeným

a v případě, že je právě vygenerované řešení lepší, tak si jej zapamatuje. Tento postup můžeme provádět dokud máme k dispozici čas, nebo dokud nenajdeme řešení dostatečné kvality.

Vstupy : $N \in \mathbb{N}$, cas, $tolerance$

Výstupy : H, množina orientovaných hran obsahující jednu hranu nulové délky, pro $i \in \mathbb{N}$ taková, že $1 \leq i < \frac{N}{2}$ H obsahuje právě jednu hranu délky i kladné i záporné orientace a pokud je N sudé, tak i jednu libovolnou orientovanou hranu délky $\frac{N}{2}$

Inicializace: $\forall i \in \left\{0, \ldots, \left\lceil \frac{N}{2} - 1 \right\rceil\right\}$, $M_i = \left\{e = (u,v) \in E\left(K_N\right) : t\left(u\right) \leq t\left(v\right) \wedge l\left(e\right) = i\right\}$, $M_{\frac{N}{2}} = \left\{e \in E\left(K_N\right) : l\left(e\right) = \frac{N}{2}\right\}$, $H = \emptyset$, $M = \emptyset$, $M^{opti} = V\left(K_N\right)$

funkce `Alg5(`M_{in}`)` **begin**

$L \leftarrow \left\{l\left(e = (u,v)\right) : u, v \in M_{in}\right\}$

if $|L| = \left\lfloor \frac{N}{2} \right\rfloor + 1$ **then**

return M_{in}

end

$C \leftarrow \left\{v \in V\left(K_N\right) : \left|\left\{l\left(e\right) : u \in M_{in},\, e = (u,v)\right\}\right| = \max_{w \in V(K_N)} \left|\left\{l\left(f\right) : u \in M_{in},\, f = (u,w)\right\}\right|\right\}$

$u \leftarrow$ **náhodný vrchol** $\in C$

$M_{in} \leftarrow M_{in} \cup \{u\}$

return `Alg5(`M_{in}`)`

end

begin

while $\left(\left|M^{opti}\right| > tolerance \wedge (\textit{máme dostatek času})\right)$ **do**

$M \leftarrow$ `Alg5(`$\emptyset$`)`

if $|M| < \left|M^{opti}\right|$ **then**

$M^{opti} \leftarrow M$

end

end

for $\forall i \in \left\{0, \ldots, \frac{N}{2}\right\}$ **do**

$H \leftarrow H \cup \left\{f \in \left\{e = (u,v) \in E\left(K_N\right) : \{u,v\} \subseteq M^{opti} \wedge e \in M_i\right\}\right\}$

end

return H

end

Algoritmus 7: Nalezení aproximace H^{opti} metaheuristicky, náhodným generováním s využitím znalosti problému.

Předpokládejme, že jsme schopni vybrat náhodný prvek z kandidátské množiny v konstantním čase, pak asymptotickou složitost vygenerování **právě jednoho** řešení algoritmem 7 můžeme odhadnout vztahem $\mathcal{O}\left(\sum_{i=1}^{\frac{N}{2}}(N-i)\,i\right)$, l náhodných řešení pak získáme v čase $\mathcal{O}\left(l\left[\sum_{i=1}^{\frac{N}{2}}(N-i)\,i\right]\right)$.

Jedním nedostatkem algoritmů 6 a 7 je například to, že i když se jim podaří vygenerovat kvalitní řešení, tak v dalším kroce generuje nové řešení opět zcela náhodně a bez jakékoliv znalosti řešení dříve nalezených. Níže tedy představujeme algoritmus 8, jenž je založen na evolučním přístupu. Algoritmus 8 nejdříve vygeneruje množinu náhodných řešení (buď algoritmem 6 nebo algoritmem 7), a posléze každé takovéto řešení iteračně upravuje tak, že v jednom kroce odstraní náhodný vrchol, poté sestaví kandidátskou množinu, jak je tomu v případě algoritmu 5, a z té náhodně vybere vrchol, který do řešení opět zařadí.

Algoritmus 8 má tedy dvě základní vlastnosti:

- "nezapomíná" kvalitní řešení.
- řešení se stále generují náhodně, evolučně.

Vstupy : $N \in \mathbb{N}$, $tolerance$, cas

Výstupy : H, množina orientovaných hran obsahující jednu hranu nulové délky, pro $i \in \mathbb{N}$ taková, že $1 \leq i < \frac{N}{2}$ H obsahuje právě jednu hranu délky i kladné i záporné orientace a pokud je N sudé, tak i jednu libovolnou orientovanou hranu délky $\frac{N}{2}$

Inicializace: $\forall i \in \left\{0, \ldots, \left\lceil \frac{N}{2} - 1 \right\rceil\right\}$, $M_i = \{e = (u,v) \in E(K_N) : t(u) \leq t(v) \wedge l(e) = i\}$, $M_{\frac{N}{2}} = \left\{e \in E(K_N) : l(e) = \frac{N}{2}\right\}$, $H = \{\emptyset\}$, $M = \{\emptyset\}$

begin

$M \leftarrow$ `Alg4(∅)` **nebo** $M \leftarrow$ `Alg5(∅)`

while ($|M| > tolerance \wedge$ (*máme dostatek času*)) **do**

$u \leftarrow$ **náhodný vrchol** $\in M$

$M \leftarrow M \setminus \{u\}$

$$C \leftarrow \left\{v \in V(K_N) : |\{l(e) : u \in M,\, e = (u,v)\}| = \max_{w \in V(K_N)} |\{l(f) : u \in M,\, f = (u,w)\}|\right\}$$

$u \leftarrow$ **náhodný vrchol** $\in C$

$M \leftarrow M \cup \{u\}$

end

for $\forall i \in \left\{0, \ldots, \frac{N}{2}\right\}$ **do**

$H \leftarrow H \cup \{f \in \{e = (u,v) \in E(K_N) : \{u,v\} \subseteq M \wedge e \in M_i\}\}$

end

return H

end

Algoritmus 8: Nalezení aproximace H^{opti} metaheuristicky, s využitím evolučního generování řešení.

Asymptotickou náročnost algoritmu 8 můžeme odhadnout následovně:

- Pokud využijeme jako generátor počátečního řešení algoritmus 6, pak počáteční řešení získáme v čase $\mathcal{O}(N)$(maximální počet prvků řešení je řádově $\frac{N}{2}$). Pokud vycházíme z předpokladu, že počáteční řešení obsahuje maximální počet, čili$\mathcal{O}\left(\frac{N}{2}\right)$, vrcholů, tak každá následná úprava počátečního řešení bude mít náročnost $\left(\frac{N}{2}\right)^2$(při generování kandidátské množiny vrcholů porovnáme $\frac{N}{2}$ prvků řešení s $\frac{N}{2}$ prvky mimo řešení). Provedení l iterací algoritmem 8 pak bude mít náročnost $\mathcal{O}\left(lN^2 + N\right) = \mathcal{O}\left(lN^2\right)$. Provedení l iterací na p vygenerovaných řešeních pak můžeme odhadnout výrazem $\mathcal{O}\left(lN^2 + pN\right)$.
- Pokud využijeme jako generátor počátečního řešení algoritmus 7, pak počáteční řešení získáme v čase $\mathcal{O}\left(\sum_{i=1}^{\frac{N}{2}} (N-i)\, i\right)$. Pokud vycházíme z předpokladu, že počáteční řešení obsa-

huje maximální počet, čili$\mathcal{O}\left(\frac{N}{2}\right)$, vrcholů, tak každá následná úprava počátečního řešení bude mít náročnost $\left(\frac{N}{2}\right)^2$. Vygenerování l náhodných řešení algoritmem 8 pak bude mít náročnost $\mathcal{O}\left(lN^2 + \sum_{i=1}^{\frac{N}{2}} (N-i)\, i\right)$. Provedení l iterací na p vygenerovaných řešeních pak můžeme odhadnout výrazem $\mathcal{O}\left(lN^2 + p\sum_{i=1}^{\frac{N}{2}} (N-i)\, i\right)$.

5 Naměřené výsledky

V předchozí kapitole jsme formulovali skupinu algoritmů generující cyklická rozdělení práce mezi procesory. V této kapitole jednotlivé algoritmy porovnáme a zanalyzujeme jejich vlastnosti samostatně i navzájem mezi sebou.

V této kapitole budeme používat výraz **kvalita řešení**, kterým budeme rozumět počet vrcholů potřebných k indukování všech hran řešení poskytnutého algoritmy uvedených v předchozí kapitole. Čím nižší tento počet vrcholů bude, tím budeme považovat řešení za kvalitnější.

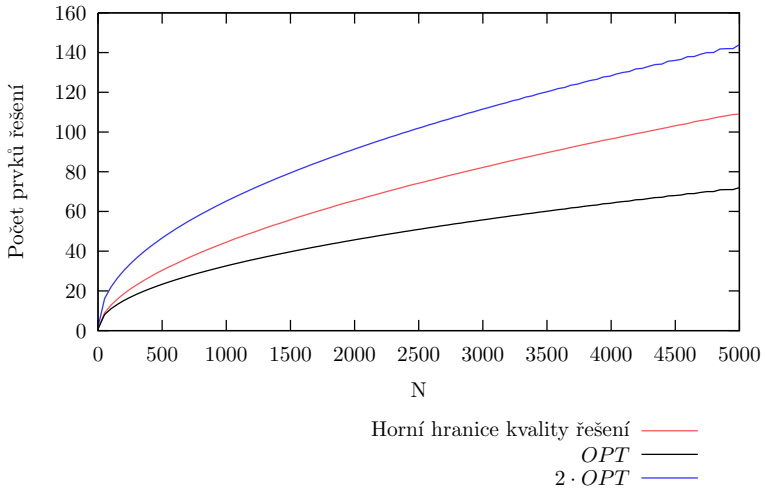
S kvalitou řešení souvisí další výraz, a sice **teoretické optimum**, kterým budeme rozumět minimální počet vrcholů potřebných k indukování všech hran poskytnutých algoritmy v předchozí kapitole. Hodnotu teoretického optima je možno odvodit na základě poznání, že v grafu K_N je $\lfloor \frac{N}{2} \rfloor$ různých délek hran, přičemž v řešení musí být obsažena alespoň jedna hrana každé délky. Minimální počet vrcholů (označme OPT) potřebných k indukování $\lfloor \frac{N}{2} \rfloor$ neorientovaných hran (za předpokladu, že bude mít každá indukovaná hrana unikátní délku) pak získáme ze vztahu vyjadřující počet hran kompletního grafu, a sice:

$$\begin{aligned} OPT\,(OPT-1)\,\frac{1}{2} &= \lfloor \tfrac{N}{2} \rfloor \\ OPT &= \left\lceil \tfrac{1}{2}\left(\sqrt{8\,\lfloor \tfrac{N}{2} \rfloor + 1} + 1\right)\right\rceil \end{aligned}$$

5.1 Analýza heuristického algoritmu

Nejdříve se zaměříme na analýzu deterministického algoritmu 5 pro různá N a s různými hodnotami parametru k.

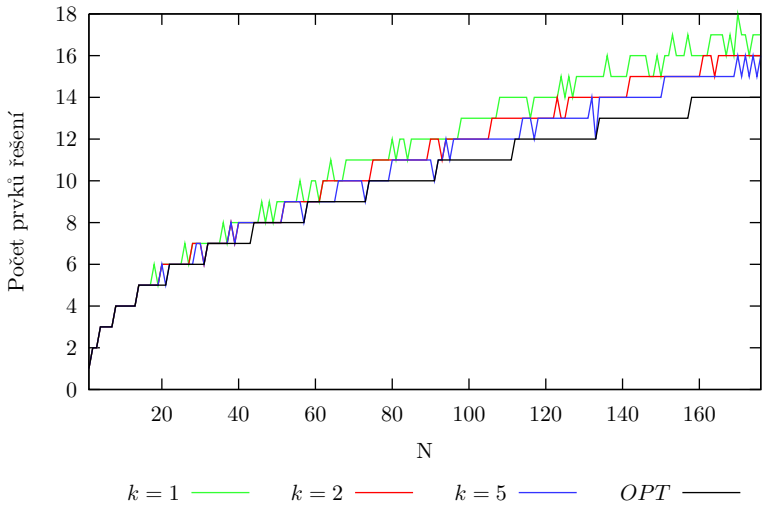
První vlastností, jež by nás mohla zajímat je **horní hranice** kvality řešení vygenerovaných algoritmem 5. Značným omezujícím faktorem algoritmu je čas potřebný k průchodu celým definičním oborem problému. Na grafu níže tedy představujeme kvalitu řešení nalezených algoritmem 5 s parametrem $k = 1$, přičemž pro referenci jsou znázorněny také dvě křivky znázorňující teoretické optimum a dvojnásobek teoretického optima.



Obrázek 15: Kvalita řešení nalezených algoritmem 5 pro $k = 1$ a $1 \leq N \leq 5000$.

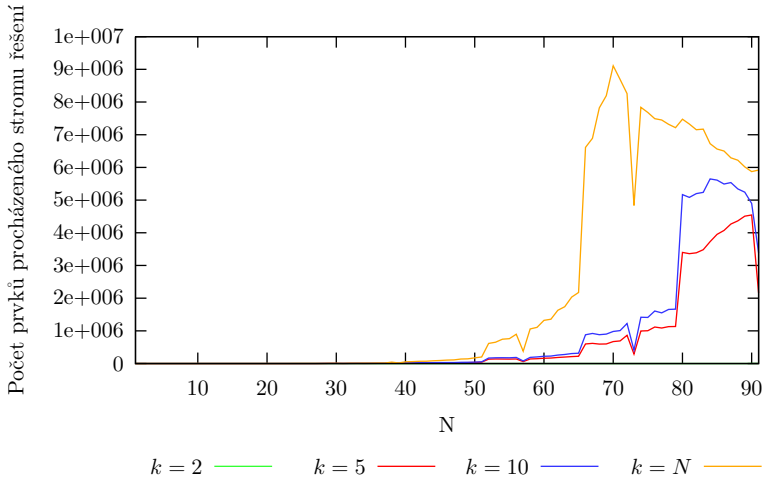
Z grafu na obrázku 15 můžeme vidět, že se kvalita řešení pohybuje zhruba kolem 1.5-násobku teoretického minima.

Další vlastností, která by nás mohla zajímat je kvalita nalezených řešení v závislosti na zvyšující se hodnotě k. Na grafu níže můžeme vidět čtyři křivky, dvě referenční (algoritmus s parametrem $k = 1$ a teoretické optimum), jednu znázorňující kvalitu řešení nalezených s parametrem $k = 2$ a jednu znázorňující kvalitu řešení nalezených s parametrem $k = 5$. Pro směrodatné zjištění závislosti byl test pro obě k spuštěn bez časového limitu. Pro N, která nejsou na grafu znázorněna již čas potřebný k prohledání všech řešení přesahoval 5 hodin a test byl tedy přerušen.



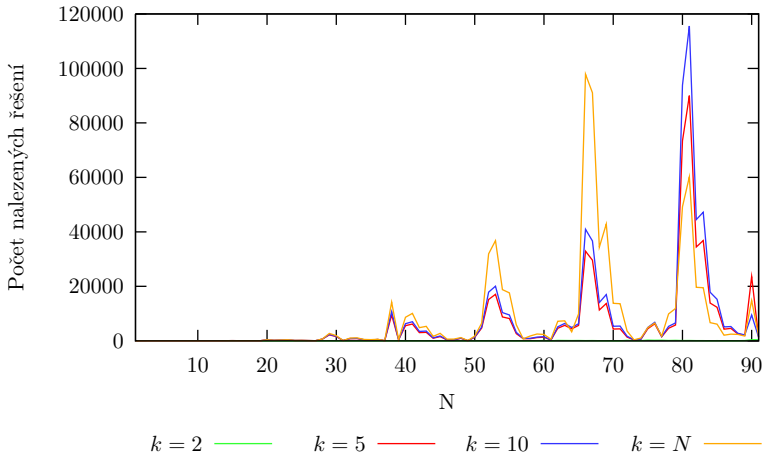
Obrázek 16: Kvality řešení nalezených algoritmem 5 pro $k \in \{1, 2, 5\}$ a $1 \leq N \leq 176$.

V části textu popisující algoritmus 5 nebyly uvedeny odhady časové složitosti (z důvodu obtížného formulování vlivu vybírání prvků z kandidátské množiny na počet prvků stromu procházených řešení). Pro získání představy o náročnosti výpočtu jsme proto měřili počet větvení algoritmu pro různé hodnoty N a k, jež jsou k vidění níže.



Obrázek 17: Počet možností navštívených algoritmem 5 pro $k \in \{2, 5, 10, N\}$ a $1 \leq N \leq 91$.

Další vlastností algoritmu jež by nás mohla zajímat je počet různých vygenerovaných řešení v závislosti na N a hodnotě parametru k. Tato vlastnost algoritmu 5 sice nevypovídá o jeho časové složitosti, nicméně by mohla být relevantní v dalšími studiu problematiky nalezení optimálních cyklických rozdělení práce.



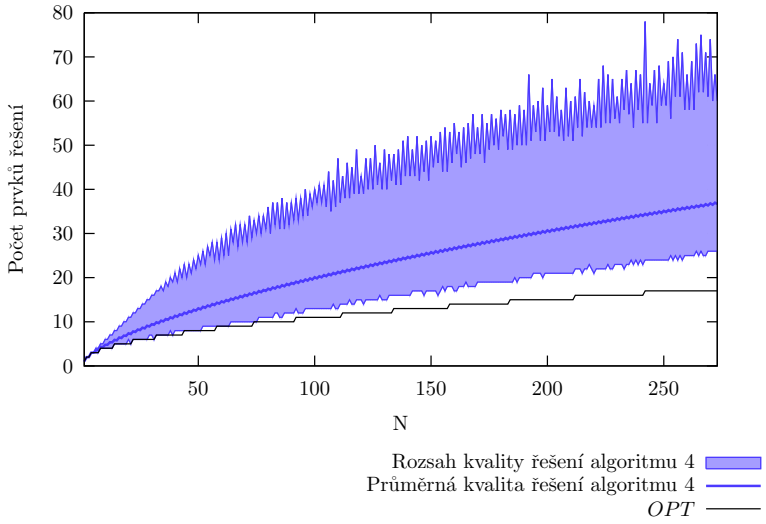
Obrázek 18: Počet nalezených řešení algoritmem 5 pro $k \in \{2, 5, 10, N\}$ a $1 \leq N \leq 91$.

Všimněme si, že pro $k = N$ a $N \approx 80$ je počet nalezených řešení nižší, než v případě $k \in \{2, 5\}$. Je to způsobeno optimalizací algoritmu, a sice takovou kdy je výpočet vrácen o krok zpět po zjištění, že momentálně generované řešení již nemůže být lepší, než prozatím nejlepší nalezené řešení.

5.2 Analýza algoritmů náhodného restartu

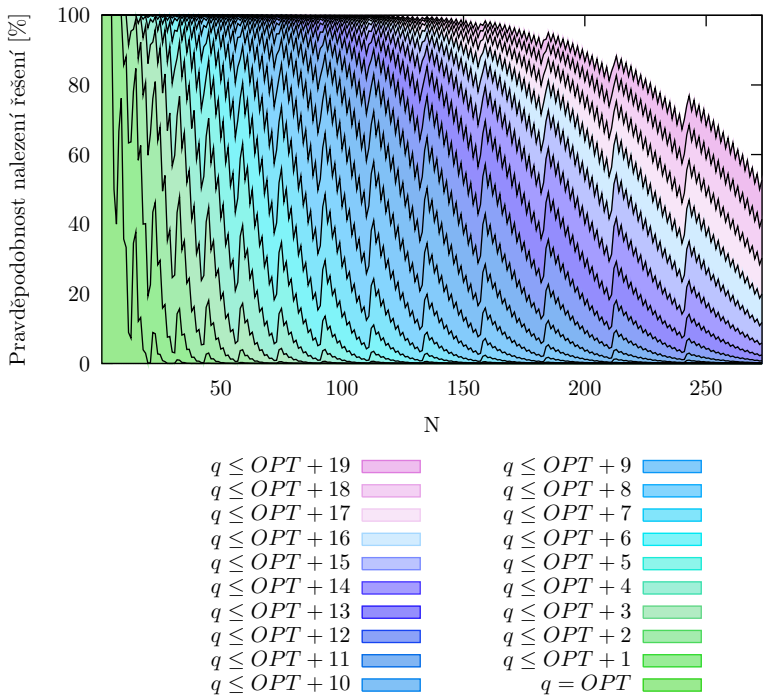
V této podkapitole se podrobněji podíváme na vlastnosti algoritmů 6 a 7. Testy proběhly pro $1 \leq N \leq 273$, přičemž pro každé N algoritmy generovaly množiny řešení po dobu 30-ti sekund.

V grafu na obrázku 19 je k vidění analýza algoritmu 6, přičemž pro každé N je k vidění minimální, maximální a průměrná kvalita z nalezených řešení. Oblast mezi minimální a maximální kvalitou řešení je vybarvena pro lepší představu o rozsahu kvality řešení.



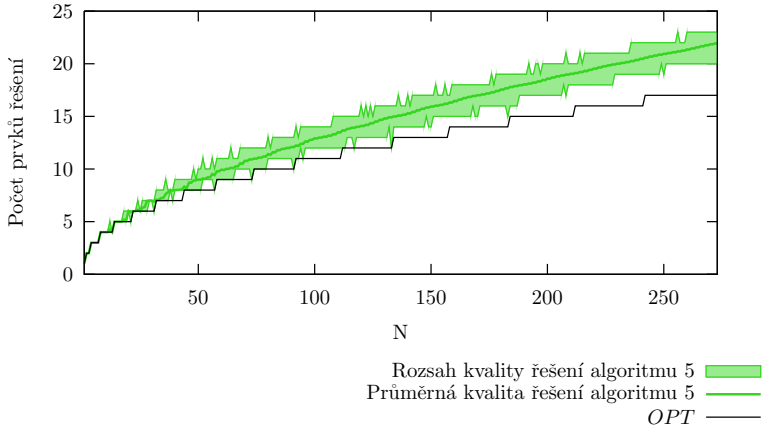
Obrázek 19: Analýza kvality řešení nalezených algoritmem 6 pro $1 \leq N \leq 273$.

Vlastností, jež by nás dále mohla zajímat je **odhad** pravděpodobnosti, že algoritmus 6 vygeneruje řešení dané kvality. V grafu na obrázku 20 jsou proto pro jednotlivá N uvedeny relativní histogramy kvality nalezených řešení. Osa x znázorňuje N, osa y znázorňuje odhad pravděpodobnosti, že algoritmus vygeneruje řešení o $q = OPT$, $q \leq OPT + 1$, $q \leq OPT + 2$, ..., $q \leq OPT + 19$ prvcích.



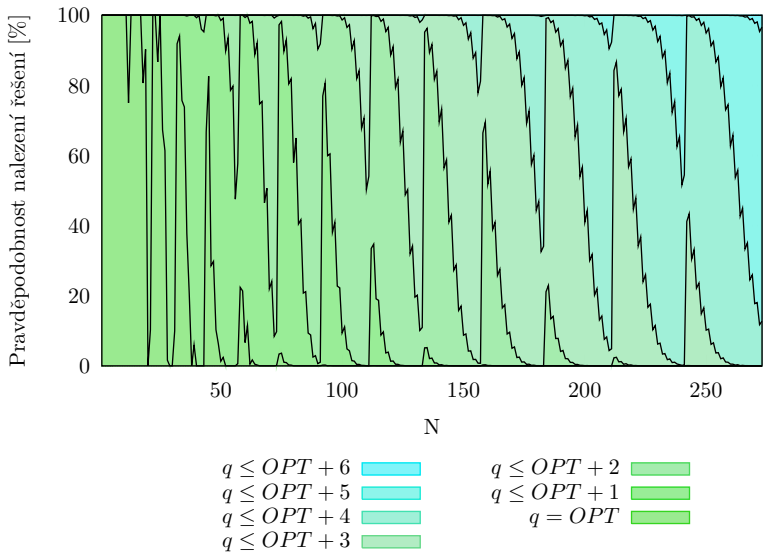
Obrázek 20: Odhad Pravděpodobnosti nalezení řešení o q prvcích algoritmem 6 pro $1 \leq N \leq 273$.

Podobným úvahám jsme rovněž podrobili algoritmus 7. V grafu na obrázku 21 je k vidění analýza algoritmu 7, přičemž pro každé N je k vidění minimální, maximální a průměrná kvalita z nalezených řešení. Oblast mezi minimální a maximální kvalitou řešení je vybarvena pro lepší představu o rozsahu kvality řešení.



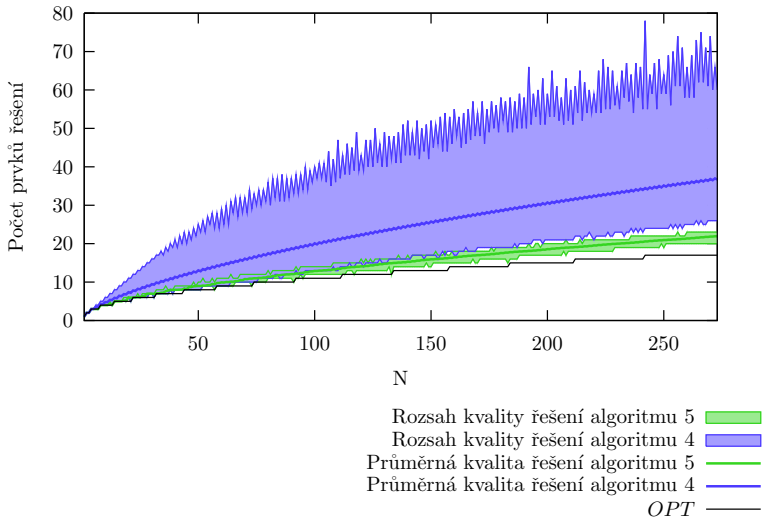
Obrázek 21: Analýza kvality řešení nalezených algoritmem 7 pro $1 \leq N \leq 273$.

V grafu na obrázku 22 jsou pro jednotlivá N uvedeny relativní histogramy kvality nalezených řešení. Osa x znázorňuje N, osa y znázorňuje odhad pravděpodobnosti, že algoritmus vygeneruje řešení o $q = OPT$, $q \leq OPT + 1$, $q \leq OPT + 2$, ..., $q \leq OPT + 19$ prvcích.



Obrázek 22: Odhad pravděpodobnosti nalezení řešení o q prvcích algoritmem 7 pro $1 \leq N \leq 273$.

Na závěr analýzy algoritmů náhodného restartu jsme oba algoritmy porovnali navzájem, viz obrázek 23.



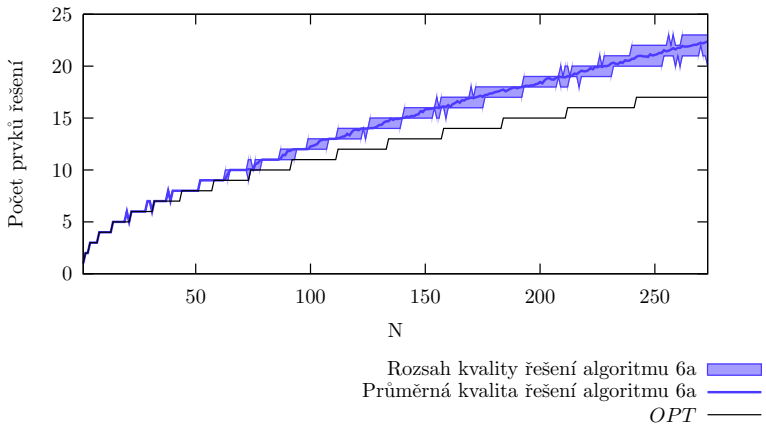
Obrázek 23: Srovnání kvality řešení vygenerovaných algoritmy 6 a 7 pro $1 \leq N \leq 273$.

Z grafu na obrázku 23 můžeme usoudit, že pokud bychom chtěli při řešení problému distribuce paměti použít pouze algoritmus náhodného restartu, pak bude vždy výhodnější využít algoritmu 7. V další části kapitoly již tedy nebudeme algoritmus 6 analyzovat.

5.3 Analýza evolučního algoritmu

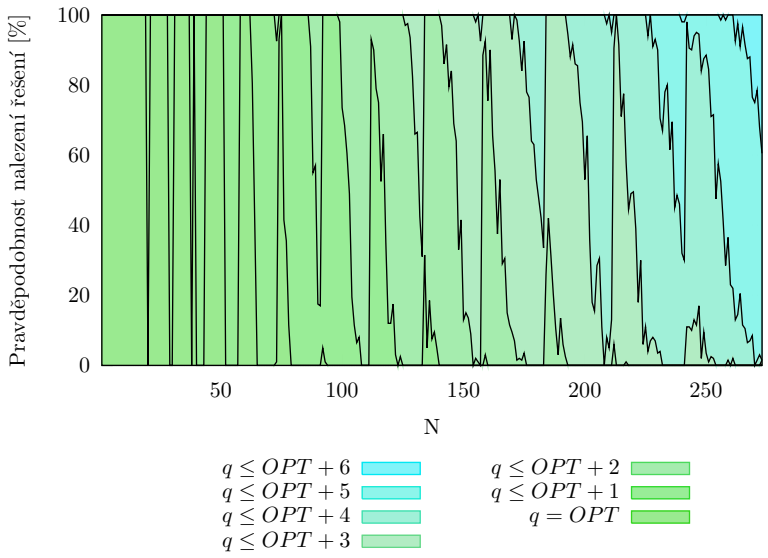
V této podkapitole se zaměříme na analýzu algoritmu 8, jež rozšiřuje algoritmy 6 či 7. Testy opět proběhly v rozsahu $1 \leq N \leq 273$, přičemž pro každé N algoritmus vygeneroval množinu 200 počátečních řešení, které ve svém průběhu upravoval po dobu 30-ti sekund. Algoritmus 8, jenž využívá pro vygenerování počátečních řešení algoritmu 6 budeme v této části textu označovat 8a. Algoritmus 8, jenž využívá pro vygenerování počátečních řešení algoritmu 7 pak budeme označovat 8b.

V grafu na obrázku 24 je k vidění analýza algoritmu 8a. Pro každé N je uvedena minimální, maximální a průměrná kvalita z množiny vygenerovaných řešení. Oblast mezi minimální a maximální kvalitou řešení je vybarvena pro lepší představu o rozsahu kvality řešení.



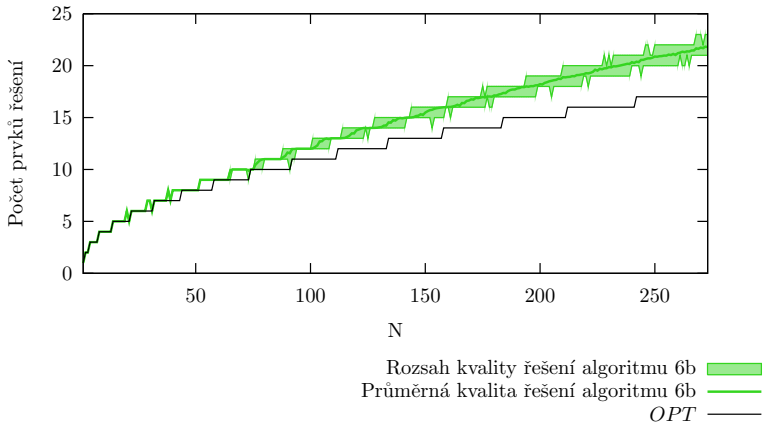
Obrázek 24: Analýza kvality řešení nalezených algoritmem 8a pro $1 \leq N \leq 273$.

V grafu na obrázku 25 jsou pro jednotlivá N uvedeny relativní histogramy kvality nalezených řešení algoritmem 8b. Osa x znázorňuje N, osa y znázorňuje odhad pravděpodobnosti, že algoritmus vygeneruje řešení o $q = OPT$, $q \leq OPT + 1$, $q \leq OPT + 2$, ..., $q \leq OPT + 19$ prvcích.



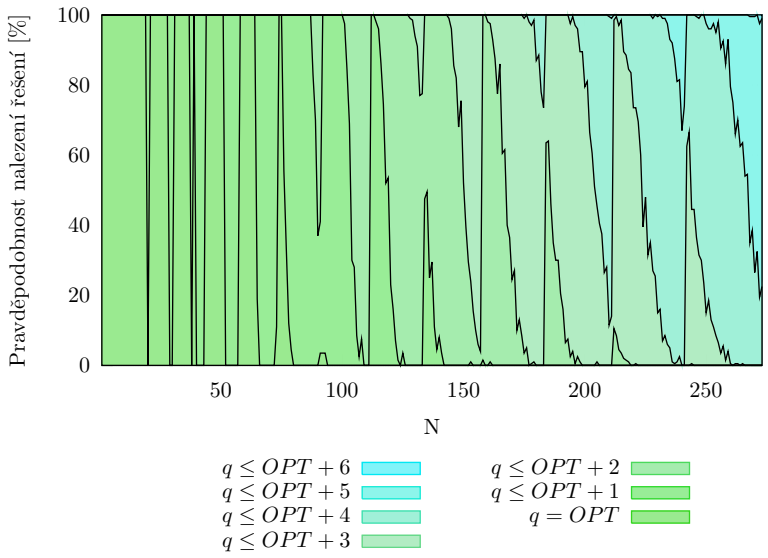
Obrázek 25: Odhad pravděpodobnosti nalezení řešení o q prvcích algoritmem 8a pro $1 \leq N \leq 273$.

Podobným úvahám jsme rovněž podrobili algoritmus 8b. V grafu na obrázku 26 je k vidění analýza algoritmu 8b, přičemž pro každé N je k vidění minimální, maximální a průměrná kvalita z nalezených řešení. Oblast mezi minimální a maximální kvalitou řešení je vybarvena pro lepší představu o rozsahu kvality řešení.



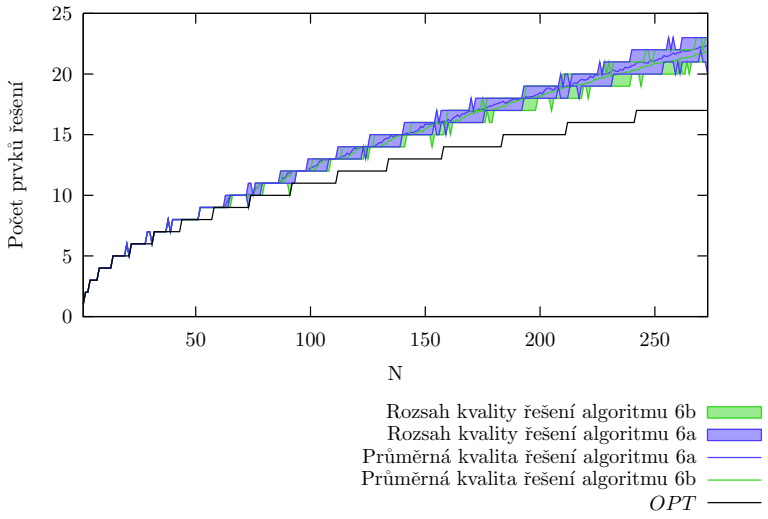
Obrázek 26: Analýza kvality řešení nalezených algoritmem 8b pro $1 \leq N \leq 273$.

V grafu na obrázku 27 jsou pro jednotlivá N uvedeny relativní histogramy kvality nalezených řešení. Osa x znázorňuje N, osa y znázorňuje odhad pravděpodobnosti, že algoritmus vygeneruje řešení o $q = OPT$, $q \leq OPT + 1$, $q \leq OPT + 2$, …, $q \leq OPT + 19$ prvcích.



Obrázek 27: Pravděpodobnost nalezení řešení o q prvcích algoritmem 8b pro $1 \leq N \leq 273$.

Na závěr analýzy evolučního algoritmu jsme oba algoritmy porovnali navzájem, viz obrázek 28.



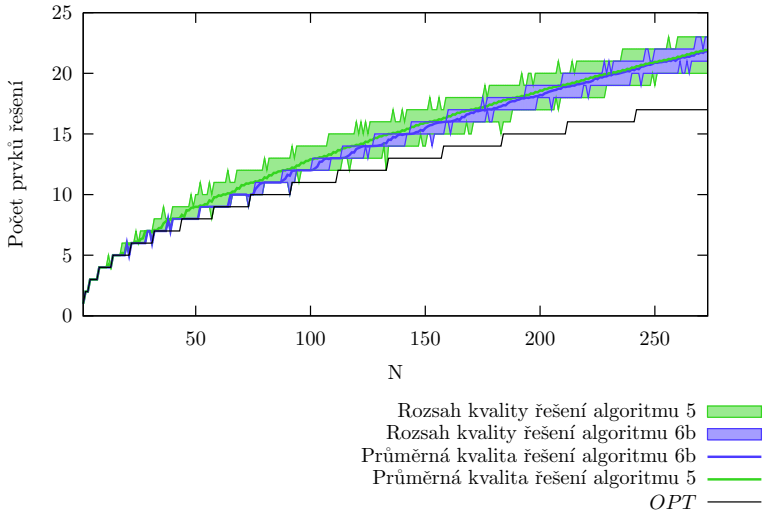
Obrázek 28: Porovnání kvality řešení vygenerovaných algoritmem 8a oproti řešením vygenerovaných algoritmem 8b pro $1 \leq N \leq 273$.

Z grafu 28 můžeme usoudit, že pokud bychom chtěli při řešení problému distribuce paměti použít evoluční algoritmus, pak bude pravděpodobně výhodnější využít algoritmu 8b. V další části kapitoly již tedy nebudeme algoritmus 8a analyzovat.

5.4 Srovnání evolučních algoritmů a algoritmů náhodného restartu

V této podkapitole porovnáme vlastnosti dvou typově různých metaheuristických algoritmů, konkrétně 7 a 8 s algoritmem 7 využitým jako generátoru počátečních řešení (z každé kategorie jsme zvolili algoritmus produkující kvalitnější výsledky).

Na grafu níže je uveden rozsah kvalit řešení vygenerovaných oběma algoritmy. Pro každé N byl časový limit generování řešení stanoven na 30 sekund. Evoluční algoritmus pracoval s 200 řešeními, která dále upravoval.



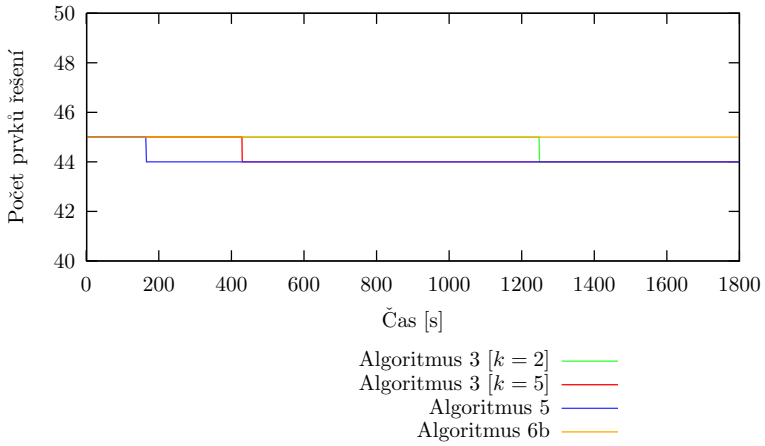
Obrázek 29: Porovnání kvality řešení vygenerovaných algoritmem 7 oproti řešením vygenerovaných algoritmem 8b pro $1 \leq N \leq 273$.

Z grafu na obrázku 29 můžeme usoudit, že algoritmus 8b má oproti algoritmu 7 nižší rozptyl kvality vygenerovaných řešení a nižší průměrnou kvalitu nalezeného řešení. Nicméně můžeme také pozorovat, že ve většině případů algoritmus 7 nalezl v poskytnutém čase kvalitnější řešení než algoritmus 8b. Tento úkaz by mohl být způsoben například velikostí množiny počátečních řešení, která algoritmus 8b dále upravuje a obecně špatnou vlastností algoritmů opustit lokální minimum.

5.5 Celkové srovnání algoritmů

Na závěr porovnáme kvalitu řešení vygenerovaných jednotlivými algoritmy v závislosti na čase. Test byl proveden pro $N = 1057$ a testovány byly následující algoritmy:

- Algoritmus 5 s parametrem $k \in \{2, 5\}$.
- Algoritmus 7.
- Algoritmus 8b.



Obrázek 30: Porovnání kvality řešení reprezentantů všech tří tříd algoritmů představených v předchozí kapitole.

Z grafu 30 je patrné, že i pro vysoká N zmíněné algoritmy produkují kvalitativně velice podobné výsledky. Usoudil bych tedy, že generování řešení způsobem popsaným v algoritmu 5, které je dále využíváno a rozšiřováno zbývajícími algoritmy, snadno konverguje do lokálních minim.

6 Závěr

V prvních dvou kapitolách tohoto textu jsme Vás obeznámili s teoretickými základy umožňující formulování dvou-dimenzionální Metody Hraničních Prvků používanou pro řešení systémů diferenciálních rovnic. Rovněž zde byla představena metoda paralelizace výpočtu využívající Richardsonovy iterační metody. Bylo představeno zamyšlení nad optimalizací paměťových nároků pro realizaci výpočtu.

Ve třetí kapitole jsme se zaměřili na problematiku nalezení takového rozdělení práce mezi procesory, aby byly nejvyšší paměťové nároky libovolného procesoru co nejnižší. Nejdříve jsme uvedli úvahy založené na triviálním přístupu a problém jsme řešeili hrubou silou. Ukázali jsme si, že pro počet procesorů vyšší než 5 nejsme schopni nalézt rozdělení práce v přijatelném čase. Proto jsme se zaměřili na hledání cyklických rozdělení práce, k čemuž jsme vybudovali aparát se základy v teorii grafů. Na základě závěrů vyplývajících z formulované teorie jsme navrhli třídu deterministických i náhodných algoritmů, jež jsou schopny nalézt taková rozdělení práce, jež jsou pro $N \leq 5000$ zhruba o 50% horší, než teoreticky nejlepší rozdělení.

Ve čtvrté kapitole jsou uvedena srovnání formulovaných algoritmů, ze kterých můžeme usoudit, že asymptotická kvalita nalezených řešení jednotlivými algoritmy je totožná.

Možné směry, jakými by bylo možno se ubírat v dalším studiu minimalizace paměťových nároků výpočtu:

- studium tzv. Perfect Difference Sets, které jsou zkonstruovatelné pro $N = p^{2n} + p^n + 1$, kde p je prvočíslo a $n \in \mathbb{N}$.
- hledat acyklická řešení, popřípadě ukázat pro jaká netriviální N nemohou být acyklická řešení lepší než řešení cyklická.
- hledat rozdělení práce, jež nevyužívá rozdělení matice A pouze do čtvercových bloků o stejných velikostech.
- hledat takové rozdělení práce, které primárně minimalizuje čas potřebný k výpočtu a současně minimalizuje potřebnou paměť.

Michal Kravčenko

7 Reference

[1] J. Bouchala: *Úvod do 'Boundary Elements Method'*, Prezentace, VŠB-TU Ostrava, Leden 2007, k dispozici na `http://homel.vsb.cz/~bou10/archiv/Bouchala_BEM.pdf`.

[2] O. Jakl: *Paralelní a Distribuované systémy*, Prezentace z přednášek, VŠB-TU Ostrava, 2008.

[3] P. Kovář: *Teorie grafů*, Skripta k výuce předmětu, VŠB-TU Ostrava, 2012, k dispozici na `http://homel.vsb.cz/~kov16/files/skriptum_teorie_grafu_rozsirene.pdf`.

[4] D. Lukáš: *Matematické modelování elektromagnetických polí*, Skripta k výuce předmětu , VŠB-TU Ostrava, 2011, k dispozici na `http://homel.vsb.cz/~luk76/Teaching/MMEP/MMEP_skripta.pdf`.

[5] D. Lukáš: *Superpočítání s vtipem*, Přednáška k výročí Katedry Aplikované Matematiky, VŠB-TU Ostrava, 2012, k dispozici na `http://lukas.am.vsb.cz/talks/talk.pdf`.

[6] M. Sadowská: *Řešení variačních nerovnic pomocí hraničních integrálních rovnic*, Diplomová práce, VŠB-TU Ostrava, 2005, k dispozici na `http://www.am.vsb.cz/theses/sadowska_ing.pdf`.