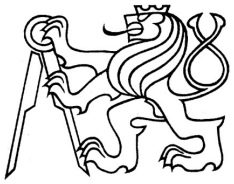
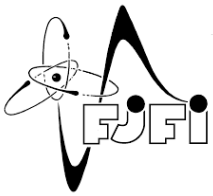


České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Náhodné algoritmy: teorie a implementace

Randomized algorithms: theory and implementation

Bakalářská práce

Autor: **Matěj Trödler**

Vedoucí práce: **doc. RNDr. Jan Vybíral, Ph.D.**

Akademický rok: 2022/2023

Katedra: matematiky Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Matěj Trödler
Studijní program:	Aplikované matematicko-stochastické metody
Název práce (česky):	Náhodné algoritmy: teorie a implementace
Název práce (anglicky):	Randomized algorithms: theory and implementation

Pokyny pro vypracování:

1) Náhodné algoritmy řeší zadaný problém za použití menšího či většího počtu náhodných bitů. V současné době nachází řadu uplatnění v diskrétní matematice, teorii grafů, strojovém učení, nebo optimalizaci.
2) Student se seznámí se základními příklady randomizovaných algoritmů a matematickým aparátem nezbytným pro analýzu jejich vlastností (odhady chyb, délka běhu, apod.)
3) Student provede analýzu několika vybraných složitějších aplikací randomizovaných algoritmů (např. problém výběrčího kuponů, expanding graphs, teorie diskrepance nebo i jiné), včetně odvození potřebných statistických odhadů.
4) Student naprogramuje pro různé parametry tyto zvolené algoritmy a demonstruje tak prakticky jejich vlastnosti při řešení zadaných problémů. Tam, kde to bude možné, provede srovnání s dostupnými deterministickými algoritmy.

Doporučená literatura:

1) N. Alon, and J. H. Spencer: The probabilistic method, Wiley-Interscience Series in Discrete Mathematics and Optimization. A Wiley-Interscience Publication, John Wiley & Sons, Inc., New York, 1992.

2) R. Motwani and P. Raghavan: Randomized algorithms, Cambridge University Press, Cambridge, 1995.

Jméno a pracoviště vedoucího bakalářské práce:

doc. RNDr. Jan Vybíral, Ph.D.
Katedra matematiky, FJFI ČVUT v Praze, Trojanova 13, 120 00 Praha

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2022

Datum odevzdání bakalářské práce: 2.8.2023

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 31.10.2022

.....
garant oboru

.....
vedoucí katedry

.....
děkan

Poděkování:

Chtěl bych zde poděkovat především svému školiteli, panu docentu Janu Vybíralovi, za pečlivost, ochotu, vstřícnost a odborné i lidské zázemí při vedení mé bakalářské práce. Zároveň patří veliké díky panu profesoru Vojtěchu Rödlovi a jeho doktorskému studentovi Marcelu Tadeu Salesovi za pomoc a nápady při snahách o vylepšení odhadů.

Čestné prohlášení:

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 25. července 2023 Matěj Trödler

Název práce:

Náhodné algoritmy: teorie a implementace

Autor: Matěj Trödler

Program: Aplikované matematicko–stochastické metody

Druh práce: Bakalářská práce

Vedoucí práce: docent Jan Vybíral, Katedra matematiky, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze

Abstrakt: Tato práce pojednává o vybraných příkladech náhodných algoritmů a ukazuje tak jejich užitečnost a jednoduchost. První kapitola se zabývá algoritmy na hledání minimálního řezu grafu. Čtenáři je důkladně představena teorie dané problematiky a podrobně vysvětlen princip algoritmu. Spočtena je úspěšnost s délkou běhu a přiložena je také implementace kódu. Dále je navrženo vylepšení a řádně porovnáno s původním algoritmem. Zjistíme, že drobnou úpravou lze vytvořit velmi efektivní algoritmus, avšak za cenu mírného nárůstu délky běhu. V druhé kapitole pak zkoumáme pojem disperze. Čtenáře zasvětíme do problematiky a představíme vybrané odhady, které lze konstruovat pomocí pravděpodobnostních metod. Hlavním výsledkem práce je pak spojení disperze s jiným kombinatorickým problémem – restriction setem. To nám umožní provést nové spodní odhady na disperzi. Kromě toho ukážeme také nové, alternativní důkazy již známých odhadů. Nakonec provedeme srovnání s aktuálně nejlepšími odhady.

Klíčová slova: Náhodné algoritmy, Min-Cut, Vylepšený Min-Cut, Disperze, Restriction set

Title:

Randomized algorithms: theory and implementation

Author: Matěj Trödler

Abstract: This paper discusses selected examples of random algorithms and thus shows their usefulness and simplicity. The first chapter deals with algorithms for finding a minimal graph cut. The reader is thoroughly introduced to the problem theory and the principle of the algorithm is explained in detail. The success rate and running time are calculated and the implementation of the code is also included. Further improvements are proposed and properly compared with the original algorithm. We find that a small modification can create a very efficient algorithm, but at the cost of slightly extending the running time. In the second chapter, we deal with the concept of dispersion. We introduce the reader into the problem and present selected estimates that can be constructed using probabilistic methods. The main result of the work is then to connect the dispersion with another combinatorial problem – the restriction set. This allows us to make new lower estimates on the dispersion. In addition, we also show new, alternative evidence of already known estimates. Finally, we make a comparison with the currently best estimates.

Key words: Randomized algorithms, Min-Cut, Improvement of Min-Cut, Dispersion, Restriction set

Obsah

Úvod

Náhodné algoritmy jsou takové algoritmy, které ve svém běhu na rozdíl od těch deterministických využívají jistý prvek náhody, respektive více či méně uniformně distribuovaných bitů. Zpravidla se jedná o pseudonáhodně generované číslo či rozhodnutí. Využívány jsou zejména tehdy, kdy deterministická cesta je příliš náročná, až nemožná. Jejich randomizace pak vede většinou k velkému zjednodušení a zrychlení. Rozlišujeme dvě skupiny náhodných algoritmů. Tou první jsou takové, které pokaždé dojdou ke správnému výsledku, například Quick-Sort na třídění pole dle velikosti. Tou druhou jsou pak algoritmy, které následkem náhodného kroku mohou při opakovaných bězích dojít k různým výsledkům, například velmi známý algoritmus Monte Carlo. Analýzou chodu se pak snažíme vyčíslit délku běhu a pravděpodobnost, že nalezený výsledek algoritmu je ten správný.

Historie náhodných algoritmů sahá do poloviny 20. let a je spojena se zkoumáním vlivu náhody ve výpočetních úlohách. Jeden z prvích příkladů konstrukce randomizovaných algoritmů lze nalézt v práci Nicholase Metropolise a Stanislawa Ulama [1], kteří vyvinuli algoritmus Monte Carlo během projektu Manhattan ve 40. letech 20. století. Tato metoda využívá generování náhodných čísel k aproximaci řešení složitých matematických problémů. Na počátku 60. let pak Tony Hoare přišel s algoritmem Quick-Sort [2], který velmi rychle, jednoduše a efektivně třídí pole dle velikosti.

V 70. letech 20. století se pak začalo náhodným algoritmům dostávat veliké pozornosti díky práci Roberta W. Floyda a Ronalda L. Rivesta [3], ve které demonstrovali zlepšení efektivity algoritmu získané pomocí randomizace. O další posun se postaral Paul Erdös například v [4, 5], který za využití pravděpodobnostních metod, jejichž výhodu demonstrujeme i v následující práci, řeší náročné kombinatorické problémy a existenci objektů zkoumaných vlastností. Tyto metody se pak velmi rozšířily a stěžejní prací, ve které autoři ukázali sílu jejich aplikací, bylo [6] od Alona Noga a Joela H. Spencera publikované poprvé v roce 1991. Vrcholem problematiky náhodných algoritmů se následně stalo v roce 1995 dílo Rajeeva Motwaniho a Prabhakara Raghavana s názvem Randomized Algorithms [7], které se stalo motivací pro tuto práci.

Cílem této bakalářské práce bude představit některé jednodušší i složitější algoritmy. Jako příklad toho poměrně jednoduchého byl vybrán algoritmus Min-Cut, který hledá minimální řez multigrafu. Základ algoritmu pochází z roku 1956, kdy Lester R. Ford, Jr. a Delbert R. Fulkerson představili svůj algoritmus na hledání maximálního toku v síti [8]. Samotný Min-Cut algoritmus byl pak poprvé sestrojen Davidem R. Kargerem v roce 1993 v [9], kde představil svůj přelomový algoritmus. Svoji velmi efektivní verzi algoritmu na hledání minimálního řezu grafu sestrojili i Mechthild Stoer a Frank Wagner [10]. Zde si však postačíme s Kargerovou verzí, která svou jednoduchostí dobře ilustruje princip náhodných algoritmů. Čtenáři bude vysvětlena teorie grafů k pochopení problematiky a následně bude seznámen s principem algoritmu. Proveden bude korektní matematický popis a podrobná analýza úspěšnosti. Dále bude představeno možné vylepšení, kterým pomocí jednoduché úpravy původního algoritmu sestrojíme již algo-

ritmus složitější. Ovšem jak záhy zjistíme, povede ke znatelně lepší pravděpodobnosti úspěchu nalezení minimálního řezu, jíž důkladně spočteme a porovnáme společně i s délkou běhu. Aby si pak čtenář mohl udělat reálný obrázek o obou algoritmech a vyzkoušet si tak jeho funkčnost, přiložíme také možné implementace kódů v jazyku Python.

Druhá kapitola se zabývá problematikou disperze. V druhé polovině 20. století matematici zkoumali, jak velký kvádr lze sestrojit, aby neobsahoval žádný z daných bodů rozmístěných v jednotkové krychli [11, 12, 13]. Deterministická konstrukce takových kvádrů je pak obsahem [14]. Zde budeme naopak odhadovat, kolik bodů rozmístěných v jednotkové krychli v závislosti na dimenzi je nutno mít, aby objem kvádrů v této krychli, který nebude žádný bod obsahovat, byl menší než zadaná hodnota. Jedná se o složitý problém, který nelze vyřešit přesně. Pomoci pravděpodobnostních metod a nejrůznějších odhadů můžeme znát pouze rozmezí, ve kterém se zkoumané veličiny pohybují. Čtenáři bude představena teorie této problematiky a bude seznámen s pravděpodobnostním důkazem odhadu disperze. Generováním uniformně rozdělených bodů můžeme totiž sestrojit množinu bodů s požadovanou disperzí. Našim cílem bádání bude zjistit, respektive odhadnout velikost této množiny. Představíme již známý důkaz, který nám pomůže spojit problematiku disperze s jiným kombinatorickým problémem zvaným restriction set. Uvedeme zde známé odhady obou veličin, pokusíme se přijít na odhady nové a porovnáme je s těmi nejlepšími.

Celkovým cílem práce je tedy uvést některé druhy náhodných algoritmů, představit je a ilustrovat tak jejich princip a výhody. Provedeme důkladnou analýzu algoritmů a, kde to bude možné, implementujeme prakticky kód algoritmu v jazyku Python.

Kapitola 1

Min-Cut Algoritmus

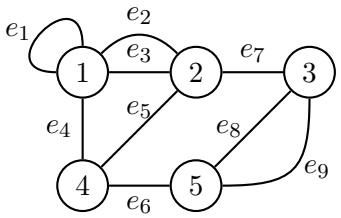
Jako první představíme a zanalyzujeme algoritmus s názvem *MinCut*. Jeho úkolem bude najít v souvislém grafu nejmenší možný řez, který rozdělí graf na dva souvislé podgrafy. Jako jednoduchý příklad využití může posloužit situace, kdy každý vrchol grafu představuje pozice protivníka a hrany mezi nimi značí počet komunikací, které je spojují. Agresora by pak zajímalo, jaký nejmenší počet bomb potřebuje na zpřetrhání sítě protivníka a odizolování části jednotek od zbytku. Praktičtější aplikace pak úloha nachází v hledání slabých míst sítí, jejich optimalizace a spolehlivost [15], ve strojovém učení při učení z dat pomocí min-cutu [16] či při segmentaci obrázků [17].

Úloha hledání minimálního řezu však není tak jednoduchá, jak by se mohlo na první pohled zdát. Analyticky je téměř neřešitelná už při použití několika desítek vrcholů. Jak ale záhy uvidíme, s použitím náhodného výběru budeme schopni předpovědět alespoň pravděpodobnost, s jakou jistotou jsme schopni nalezený řez označit za skutečně minimální.

1.1 Teorie grafů

K pochopení látky nejprve uveďme základní pojmy z teorie grafů. Pracovat budeme s multigrafem, který je složen z vrcholů a hran je spojující. Následně si uvedeme, co vlastně znamená samotný pojem řez grafu. Celý algoritmus hledání minimálního řezu pak bude fungovat na principu zkracování hran, jejíž definice je uvedena v závěru této sekce.

Definice 1.1. Graf $G = (V, E)$ nazveme neorientovaným multigrafem o n vrcholech a N hranách, jestliže $V = \{U_1, \ldots, U_n\}$ je množina vrcholů $U_1, \ldots, U_n$ a $E = (e_1, \ldots, e_N)$ je posloupnost hran spojující jednotlivé vrcholy, kde $\forall i \in \hat{N},\ \exists A_i, B_i \in V : e_i = \{A_i, B_i\}$.

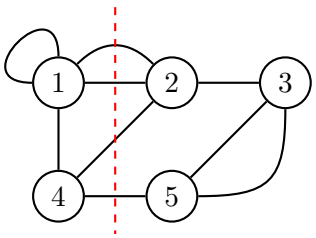


Obrázek 1.1: Příklad multigrafu o pěti vrcholech.

Poznámka 1.2. Je důležité si z definice uvědomit, že libovolné dva vrcholy může spojovat libovolný počet hran.

Definice 1.3. Graf $G = (V, E)$ nazveme souvislý, jestliže pro každé jeho dva vrcholy $A, B \in V$ existuje cesta z A do B. Třídu všech souvislých grafů označme pro jednoduchost zápisu jako $\mathcal{S}$.

Definice 1.4. Řezem grafu $G = (V, E)$ rozumíme délku takové vybrané posloupnosti F z E, že graf $G_0 = (V, E \setminus F)$ není souvislý. Číslo $k := \min_{F:\,(V,E\setminus F)\notin\mathcal{S}} |F|$ nazveme minimálním řezem grafu G neboli *min-cutem*.



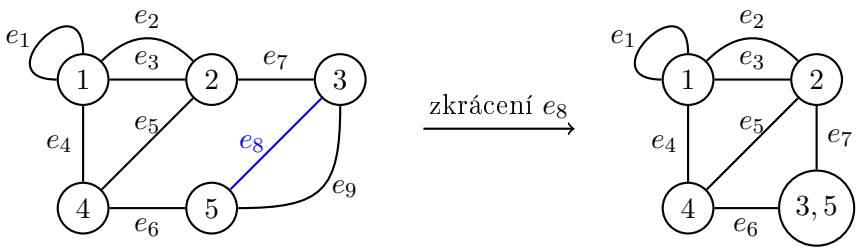
Obrázek 1.2: Řez grafu velikosti čtyři, přičemž červeně jsou přeříznuty hrany tvořící daný řez.

Poznámka 1.5. Jiná definice řezu je možná k nahlédnutí například v [6].

Definice 1.6. Nechť $G = (V, E)$ je graf o n vrcholech a N hranách, kde $V = \{U_1, \ldots, A_i, \ldots, B_i, \ldots, U_n\}$ je množina vrcholů a $E = (e_1, \ldots, e_N)$ posloupnost hran. Zkrácením hrany $e_i = \{A_i, B_i\} \in E$, kde $i \in \hat{N}$ a $A_i, B_i \in V$, přičemž $A_i \neq B_i$, rozumíme vytvoření nového grafu $G_0 = (V_0, E_0)$, kde $V_0 = \{U_1, \ldots, A_i \cup B_i, \ldots, U_n\}$ neboli $|V_0| = n - 1$ a pro posloupnost hran E_0 platí:

$$\big(\forall e \in E\big)\big(\forall U_j, U_k \in V \setminus (A_i \cup B_i)\big) : e = \begin{cases} \{A_i, B_i\} & \Rightarrow \quad e \notin E_0, \\ \{A_i, A_i\} \vee \{B_i, B_i\} & \Rightarrow \quad e \notin E_0, \\ \{A_i, U_j\} \vee \{B_i, U_j\} & \Rightarrow \quad \{A_i \cup B_i, U_j\} \in E_0, \\ \{U_j, U_k\} & \Rightarrow \quad e \in E_0. \end{cases}$$

Poznámka 1.7. Zadefinovat bychom mohli též zkracování tzv. smyčky, tedy hrany začínající a končící v témže vrcholu. Avšak takovou hranu by bylo zbytečné zkracovat, protože jak si rozmyslíme v následující sekci, smyčky velikost minimálního řezu neovlivní, proto s nimi ani nebudeme pracovat a na začátku algoritmu je všechny odstraníme.



Obrázek 1.3: Zkrácení hrany e_8.

1.2 Popis algoritmu

Úmluva 1.8. V celé následující kapitole uvažujme graf $G = (V, E)$ vždy o $|V| = n$ vrcholech a $|E| = N$ hranách.

Celý algoritmus bude fungovat na následujícím principu. Mějme graf $G = (V, E)$. Náhodně zvolíme hranu $\{A, B\} \in E$ a zkrátíme ji. Tím vzniká nový graf $G_1 = (V_1, E_1)$, který už nemá žádnou hranu $\{A, B\}$ a namísto dvou vrcholů A a B už obsahuje pouze vrchol $A \cup B$. Na graf G_1 opět aplikujeme stejný postup a takto budeme postupně tvořit poloupnost grafů dále, až nakonec sestrojíme graf G_{n-2} pouze o dvou vrcholech a nějakými hranami pouze mezi nimi. Když pak tyto hrany grafu G_{n-2} odstraníme, vzniknou dvě osamocené množiny bodů neboli nesouvislý graf, tudíž jsme našli řez grafu G_{n-2}.

Nyní je však třeba rozmyslet dvě zásadní otázky. Tou první je, jestli nalezený řez grafu G_{n-2} nějak souvisí s řezem grafu G, protože to jsou dva zcela odlišné grafy. Náš postup spočíval v tom, že zkrátíme nějakou hranu $\{A_i, B_i\}$ grafu G_i a sloučíme tyto dva vrcholy do jednoho. Nyní už v grafu G_{i+1} není poznat, kolik hran mezi A_i a B_i bylo. Jisté ale je, že existovala alespoň jedna hrana $\{A_i, B_i\}$, která v E_i byla, tudíž spojené určitě byly. Neboli postupnou transformací původního grafu G jsme došli ke grafu G_{n-2}, který ale představuje jakési „zjednodušení“ původního grafu G. Když tedy provedeme řez v grafu G_{n-2}, tak můžeme řez stejné velikosti provést i v grafu G, tudíž tímto algortimem můžeme slutečně najít řez grafu G.

Druhou otázkou je korektnost naší definice zkracování. Pokud jsme totiž zkrátili jistou hranu $\{A_i, B_i\}$ grafu G_i, ale těchto hran bylo v E_i více, tak v nově vzniklém grafu G_{i+1} už žádná z potenciálně dalších stejných hran nebude (stejně jako možné hrany $\{A_i, A_i\}$ či $\{B_i, B_i\}$). Musíme však rozmyslet, jestli tento krok neovlivní velikost minimálního řezu a tím pádem princip celého algoritmu. Jednoduchou úvahou ale zjistíme, že pokud jsou v grafu nějaké hrany začínající a končící ve stejném bodě, tzv. smyčky, tak jejich odstraněním minimální řez nezmenšíme. Pokud totiž chceme při řezání grafu přetnout i tyto smyčky, tak si tím daný řez akorát zvětšíme, protože můžeme určitě provést řez, ve kterém nebudou. Tudíž odstranění smyček během našeho zkracování vede pouze ke zjednodušení celého grafu, které velikost *min-cutu* nezmění.

Algoritmus 1: MinCut

$G_0 = (V_0, E_0)$;
$V_0 = \{U_1, \ldots, U_n\}$, $E_0 = (e_1, ..., e_N)$;
odstraň všechny smyčky v E_0;
for $1 \leq i \leq n - 2$ **do**
    náhodně vyber hranu $e = \{A, B\}$ a zkraťji;
    $E_i = E_{i-1}$ bez všech hran $\{A, B\}$ a všechny vrcholy A a B nahraď $A \cup B$;
    $V_i = V_{i-1}$ bez vrcholů A a B, ale s vrcholem $A \cup B$;
return $|E_{n-2}|$

Nakonec uvažme, že pokud jsme algoritmus konstruovali tak, že jsme hrany vybírali zcela náhodně, je zřejmé, že někdy můžeme zkrátit hranu z minimálního řezu, a tudíž námi nalezený řez nemusí být zdaleka tím minimálním. Níže se tedy budeme snažit odhadnout pravděpodobnost, s jakou můžeme nalezený řez označit za minimální, a pokud necháme algoritmus běžet vícekrát po sobě a najdeme několik řezů, s jakou pravděpodobností můžeme ten nejmenší z nich označit opravdu za *min-cut*.

1.3 Matematický popis

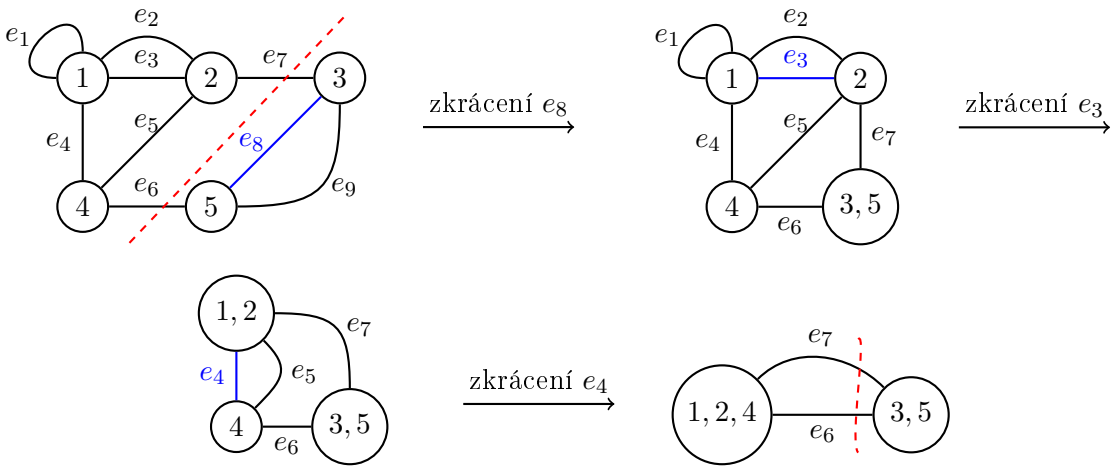
Mějme multigraf $G_0 = (V_0, E_0)$ s vrcholy $V_0 = \{U_1, \ldots, U_n\}$ a hranami $E_0 = (e_1, \ldots, e_N)$. Označme posloupnost hran jeho minimálního řezu $F = (f_1, \ldots, f_k)$, která je vybraná z E_0. Náhodně vybereme hranu $e_1 = \{A_1, B_1\}$, kde rozumíme $e_1 \in E_0$ a $A_1, B_1 \in V_0$ a zkrátíme ji. Tím vzniká nový graf $G_1 = (V_1, E_1)$.

V druhém kroku vybereme náhodně hranu $e_2 = (A_2, B_2)$ z E_1, kde $A_2, B_2 \in V_1$ a zkrátíme ji. Vzniká nový graf $G_2 = (V_2, E_2), |V_2| = n - 2$.

Takto budeme postupovat nadále neboli bude obecně pro i-tý krok $3 \leq i \leq n-2$ platit: náhodně vybereme hranu $e_i = \{A_i, B_i\} \in E_{i-1}$ a zkrátíme ji. Vzniká nový graf $G_i = (V_i, E_i)$ a $|V_i| = n - i$.

Postupným $n-2$ zkracováním dojdeme ke grafu $G_{n-2} = (V_{n-2}, E_{n-2})$ s vrcholy $V_{n-2} = \{U_{m_1} \cup \cdots \cup U_{m_p},\ U_{m_{p+1}} \cup \cdots \cup U_{m_n}\}$ a hranami $E_{n-2} = (e_{l_1}, \ldots, e_{l_q})$, přičemž jsme označili $m_1, \ldots, m_n$ permutaci na n.

Teď už máme graf pouze o dvou vrcholech a q hranách mezi nimi. Když tedy odstraníme dané hrany, vzniknou nám dva souvislé podgrafy a hrany $e_{l_1}, \ldots, e_{l_q}$ tvoří řez grafu G_{n-2} o velikosti q. Z předešlé úvahy tedy můžeme udělat řez o stejné velikosti i v grafu G_0.



Obrázek 1.4: Příklad zkracování hran grafu, dokud nezbývá graf o dvou vrcholech. Hrany e_6 a e_7 pak tvoří řez velikosti dva i v původním grafu.

1.4 Odhad řešení

V následující sekci budeme zkoumat, s jakou pravděpodobností bude $q = k$, kde jsme označili velikost *min-cutu* rovnou číslu k, popřípadě po kolika opakováních algoritmu najdeme s jistou pravděpodobností takovou posloupnost hran, že $q = k$. Předtím si však ještě uveďme větu z míry a pravdepodobnosti, tzv. součinové pravidlo.

Věta 1.9. *Nechť $\mathcal{A}$ je σ-algebra jevů, $A_1, \ldots, A_n \in \mathcal{A}$ a $P\Big(\bigcap_{i=1}^{n-1} A_i\Big) > 0$. Pak*

$$P\Big(\bigcap_{i=1}^{n} A_i\Big) = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_1 \cap A_2) \cdot \cdots \cdot P\Big(A_n \Big| \bigcap_{i=1}^{n-1} A_i\Big). \tag{1.1}$$

Důkaz. Provedeme indukcí s předpokladem, že vztah platí pro n.

- $n = 2$: $P(A_1 \cap A_2) = P(A_1) \cdot P(A_2|A_1)$ je samotný definiční vztah podmíněné pravděpodobnosti.
- $n \to n+1$: $P(A_1 \cap \cdots \cap A_n \cap A_{n+1}) = P\Big(\bigcap_{i=1}^{n} A_i\Big) \cdot P\Big(A_{n+1}|\bigcap_{i=1}^{n} A_i\Big)$, kde víme, že vztah pro $P\Big(\bigcap_{i=1}^{n} A_i\Big)$ platí z indukčního předpokladu. Tím je důkaz dokončen.

□

Nyní již přistoupíme k samotnému odvození a odhadu pravděpodobnosti, že námi nalezený minimální řez je opravdu *min-cutem* daného grafu [7]. Jelikož se jedná o jednoduchý algoritmus, tak úspěšnost nebude příliš dobrá, o to snažší však bude odvození úspěšnosti.

Věta 1.10. *Nechť $G = (V, E)$ je multigraf, C je posloupnost hran minimálního řezu vybraná z E, $|C| = k$ a A_i značí událost, že v i-tém kroku algoritmu nezkrátíme hranu z C. Pak*

$$P\Big(\bigcap_{i=1}^{n-2} A_i\Big) \geq \frac{2}{n^2}.$$

Důkaz. Na začátku je třeba si uvědomit, že jestliže *min-cut* obsahuje k hran, tak každý vrchol grafu G musí mít alespoň k sousedů, neboť kdyby jich měl méně, stačilo by odstranit pouze tyto hrany je spojující a potom by $|C| < k$, což by byl spor s předpokladem. Proto G má nejméně $\frac{k \cdot n}{2}$ hran (dvojkou dělíme z jednoduchého důvodu, že všechny hrany jsou započteny dvakrát).

Nyní odhadneme zespodu pravděpodobnost, že žádná hrana z C není během postupného zkracování vybrána. Potom hrany, které zůstanou nezkráceny po $n-2$ krocích, budou právě hrany z C. Nechť při prvním zkracování vybereme hranu $e_{j_1} = (u_1, v_1)$. Pravděpodobnost, že $e_{j_1} \notin C$, spočteme pomocí komplementárního jevu

$$P(A_1^C) \leq \frac{k}{\frac{k \cdot n}{2}},$$

jelikož počet hran grafu může být větší. Odtud ze základního axiomu pravděpodobnosti

$$P(A) = 1 - P(A^C) \tag{1.2}$$

dostaneme již pravděpodobnost $P(A_1) \geq 1 - \frac{2}{n}$.

Pokud jsme tedy v prvním kroku žádnou hranu z C nevybrali, v druhém kroku vybereme náhodně hranu $e_{j_2} = (u_2, v_2)$ a budeme se ptát na pravděpodobnost, že $e_{j_2} \notin C$. Pro doplněk jevu A_2 platí

$$P(A_2^C|A_1) \leq \frac{k}{\frac{k(n-1)}{2}},$$

neboť G_1 už obsahuje jen $n-1$ vrcholů. Neboli $P(A_2|A_1) \geq 1 - \frac{2}{n-1}$ a vzniká graf $G_2 = (V_2, E_2)$, $|V_2| = n - 2$.

Tímto způsobem budeme zkracovat hrany i nadále. V algoritmu můžeme najít jednoduchou závislost. V každém i-tém kroku máme $n - i + 1$ vrcholů, stále platí $|C| = k$, tedy celkem graf obsahuje v daný moment vždy alespoň $\frac{k(n-i+1)}{2}$ hran. Můžeme tedy konstatovat, že

$$P\Big(A_i|\bigcap_{j=1}^{i-1} A_j\Big) \geq 1 - \frac{2}{n-i+1}, \quad \forall i \in \{1, \ldots, n-2\}.$$

Nyní již můžeme pomocí vztahu (1.1) spočítat pravděpodobnost, že během $n-2$ zkracování hran nevybereme žadnou hranu z minimálního řezu. Neboli

$$P\Big(\bigcap_{j=1}^{n-2} A_j\Big) \geq \prod_{i=1}^{n-2}\Big(1-\frac{2}{n-i+1}\Big) = \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdot\ldots\cdot\frac{3}{5}\cdot\frac{2}{4}\cdot\frac{1}{3} = \frac{2}{n(n-1)}, \quad (1.3)$$

což ještě můžeme odhadnout

$$P\Big(\bigcap_{j=1}^{n-2} A_j\Big) \geq \frac{2}{n^2}. \quad (1.4)$$

□

Výše odvozeným vzorečkem jsme tedy pravděpodobnost, že v žádném z $n-2$ kroků algoritmu nezkrátíme hranu z $|C|$, zespod omezili velmi malým číslem. Proto spíše najdeme řez o velikosti větší než k neboli $q > k$. Intuitivně nám je tedy jasné, že čím více budeme daný algoritmus opakovat a z nalezených řezů vybereme ten nejmenší, s o to větší pravděpodobností o něm můžeme prohlásit, že je daným *min-cutem* grafu G. O tom pojednává následující věta.

Věta 1.11. *Nechť $G = (V, E)$ je multigraf a událost B_i značí, že v i-tém běhu algoritmu nenajdeme minimální řez. Pak po $\left\lceil \frac{n^2}{2} \right\rceil$ opakováních platí*

$$P\left(\bigcap_{i=1}^{\left\lceil \frac{n^2}{2} \right\rceil} B_i\right) < \frac{1}{e}.$$

Důkaz. Víme, že po jednom běhu daného algoritmu najdeme minimální řez s pravděpodobností alespoň $\frac{2}{n^2}$, viz vztah (1.4) neboli

$$P(B_1^C) \geq \frac{2}{n^2}.$$

Odtud můžeme pomocí (1.2) vyjádřit

$$P(B_1) \leq 1 - \frac{2}{n^2}.$$

V souladu se značením z předchozího důkazu můžeme též psát

$$B_1^C = \bigcap_{j=1}^{n-2} A_j.$$

Vzhledem ke skutečnosti, že náš algoritmus vybírá a zkracuje hrany zcela náhodně dle uniformního rozdělení neboli výsledky nalezeného řezu na konci každého běhu jsou navzájem nezávislé, a tedy i všechny jevy $B_1, \ldots, B_{\left\lceil \frac{n^2}{2} \right\rceil}$ jsou navzájem nezávislé, můžeme použít definici nezávislosti jevů a psát

$$P\left(\bigcap_{i=1}^{\left\lceil \frac{n^2}{2} \right\rceil} B_i\right) = \prod_{i=1}^{\left\lceil \frac{n^2}{2} \right\rceil} P(B_i) \leq \Big(1-\frac{2}{n^2}\Big)^{\left\lceil \frac{n^2}{2} \right\rceil} \leq \Big(1-\frac{2}{n^2}\Big)^{\frac{n^2}{2}} < \frac{1}{e},$$

neboť $P(B_i) = P(B_1)$ pro každé $i \in \{1, \ldots, \left\lceil \frac{n^2}{2} \right\rceil\}$. Ostrá nerovnost pak plyne z poznatků matematické analýzy, kdy posloupnost

$$\left(\left(1 - \frac{2}{n^2}\right)^{\frac{n^2}{2}} \right)_{n=1}^{+\infty}$$

konveruje k číslu $1/e$ zespodu. □

A co vlastně tato věta říká? Čím častěji budeme algoritmus hledání minimálního řezu opakovat, zapisovat si výsledky našeho právě nalezeného řezu a nakonec z něj vybereme ten nejmenší, tím spíše je opravdu daný řez *min-cutem* grafu $G = (V, E)$.

1.5 Kód

Tento algortimus lze implementovat tak, že vstupem bude graf ve formě struktury *dictionary*, kde *key name* je název vrcholu a samotné *items* jsou vrcholy, které jsou s daným klíčem spojené. Pak už jen náhodně volíme hrany a zkracujeme je dle definice 1.6.

```
1  import random
2
3  # Function removing self-loops
4  def remove_self_loops(graph):
5      for node in graph:
6          graph[node] = [neighbor for neighbor in graph[node] if
               neighbor != node]
7
8  def min_cut(graph):
9      """
10     The input is a dictionary that represents the graph in the
           following format: {node1: [neighbor1, neighbor2, ...], node2:
           [neighbor1, neighbor2, ...], ...}
11     """
12     remove_self_loops(graph)
13
14     while len(graph) > 2:
15         # Choose a random edge
16         node1 = random.choice(list(graph.keys()))
17         node2 = random.choice(graph[node1])
18         # Contract it
19         for w in graph[node2]:
20             # Remove edges that are the same as the contracted one
21             if w == node1:
22                 graph[w].remove(node2)
23             # Else put all other edges of node2 to node1 and set new
                   neighbors of all other vertices w
24             else:
25                 graph[node1].append(w)
26                 graph[w].remove(node2)
```

```
27                      graph[w].append(node1)
28              del graph[node2]
29
30      # Return the size of the cut
31      node1 = list(graph.keys())[0]
32      return len(graph[node1])
33
34  def repeated_min_cut(graph, repetition=100):
35      # Create empty array with cuts found in each run
36      cuts = []
37      for i in range(repetition):
38          # Find the size of the cut with min_cut and append it to the
                array
39          cut = min_cut(graph)
40          cuts.append(cut)
41      # Return the minimal cut found during all repetition
42      return min(cuts)
```

1.6 Vylepšení algoritmu

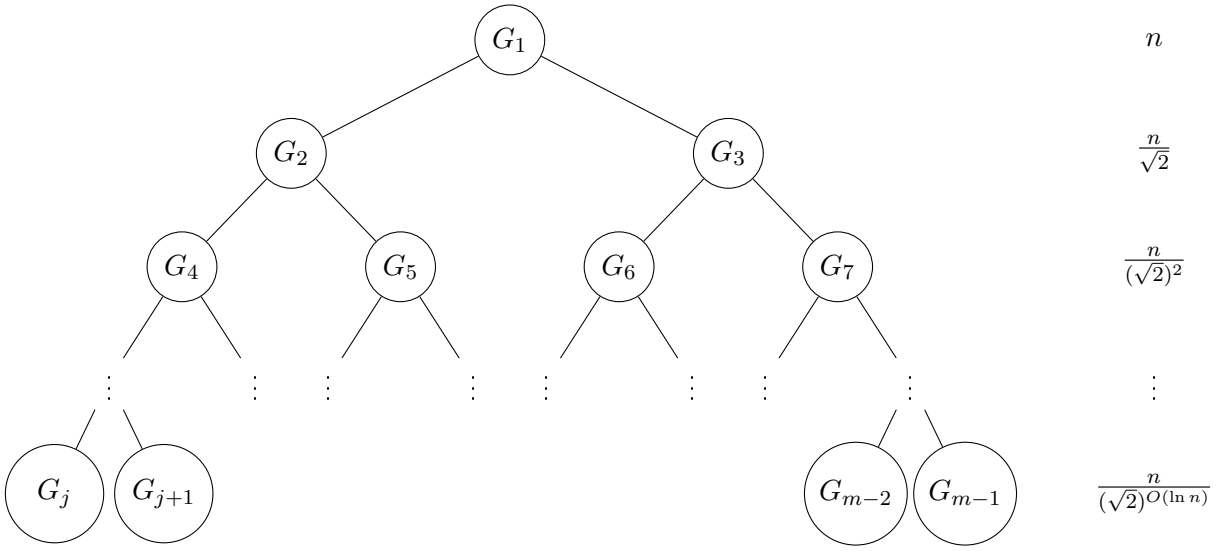
Označíme-li řád čísla jako Ω, z věty 1.10 při označení jevu $A = \bigcap_{i=1}^{n-2} A_i$ můžeme psát $P(A) = \Omega\left(\frac{1}{n^2}\right)$. To znamená, že algoritmus musíme, jak jsme ostatně zjistili ve větě 1.11, opakovat řádově $\Omega(n^2)$, abychom došli k rozumné pravděpodobnosti, že jsme našli skutečně minimální řez grafu. Je nám proto intuitivně jasné, že algoritmus *MinCut* je sice velmi jednoduchý, ale to se bohužel odráží v nízké úspěšnosti nalezení *min-cutu* a velký počet opakování zase představuje obrovskou zátěž na výpočetní techniku.

Zamysleme se proto nad tím, jak tento algoritmus vylepšit, respektive v čem tkví tak mizivá přesnost výsledku. Důvod je jednoduchý – algoritmus vybírá hrany dle uniformního rozdělení, proto když hned zkraje náhodou odstraní hranu z minimálního řezu, tak chyba zůstane až do konce algoritmu nebo se ještě dokonce zvětší.

Existuje proto nespočet lepších algoritmů, které zlepšují jak přesnost nalezeného minimálního řezu, tak i délku běhu. My si však postačíme s již stávajícím, pouze ho jen trochu modifikujeme a sestrojíme algoritmus *FastMinCut* [7]. Princip je jednoduchý. Abychom zabránili chybě již v několika prvních zkráceních náhodných hran, budeme hrany odstraňovat tak, aby se počet vrcholů snížil, ale ne o tolik. Na tento nově vzniklý graf pak pustíme sice pomalejší algoritmus, který nám ale najde minimální řez s větší pravděpodobostí než obyčejný *MinCut* a zabere méně času, než kdybychom několikrát opakovali prostý *MinCut* stále dokola.

1.6.1 Matematický popis

Mějme graf $G = (V, E)$. Na začátku pustíme na graf $G_1 := G$ dvakrát algoritmus *MinCut*, ale s tím rozdílem, že oba běhy zastavíme ve chvíli, kdy se počet vrcholů zredukoval na $n \to \left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil$. Na tyto dva nově vzniklé grafy pustíme znovu opět to samé neboli na graf G_i o t vrcholech pustíme dvakrát *MinCut* tak, aby nám vznikly dva grafy G_{2i} a G_{2i+1}, oba o $\left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil$ vrcholech. Problém nastane, když na konci budou grafy obsahovat nanejvýš 6 vrcholů. Na ně již nebudeme moci



$$j = 2^{O(\ln n)},\ m = 2j$$

Obrázek 1.5: Binární strom s grafy

pustit rekurzi, protože pro $1 \leq t \leq 6$, $t \in \mathbb{N}$: $\left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil \not< t$. V té chvíli pak již budeme muset samotný minimální řez dopočítat jinak.

Pro tak malý počet vrcholů už by bylo nejlepší spočítat minimální řez hrubou silou, neboť už to není nijak obtížná úloha. My ovšem namísto hrubé síly užijeme opět algoritmus *MinCut*, který nám vytvoří graf o dvou vrcholech a částečný *min-cut* už se najde stejně jako v původním případě – počet hran mezi těmito dvěma vrcholy tvoří řez původního grafu.

Algoritmus 2: FastMinCut

$G = (V, E)$;
Function `Rekurze`$(G_i = (V, E))$**:**
  $|V| = t$;
  if $t \leq 6$ **then**
    spočítej řez hrubou silou;
    ulož velikost řezu;
  else
    $t \leftarrow \left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil$;
    algoritmem *MinCut* vytvoř z grafu G_i dva grafy G_{2i} a G_{2i+1}, oba o t vrcholech;
    `Rekurze`(G_{2i}), `Rekurze`(G_{2i+1});
End Function;
`Rekurze`(G);
return nejmenší z nalezených řezů

Naše přemýšlení nám v našem následujícím oddíle usnadní uvědomění si, že celý algoritmus nápadně připomíná binární strom. Jeho kořen tvoří vstupní graf. Každý uzel pak představuje nově vzniklý podgraf, jehož oba synové vznikli nezávisle na sobě puštěním algoritmu *MinCut*

na svého otce (graf o t vrcholech) a obsahují pouze $\left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil$ vrcholů.

1.6.2 Analýza vylepšení

Nyní se pokusíme zjistit, jak je vlastně časově náročný algoritmus *MinCut*, a následně jej srovnáme s algoritmem *FastMinCut*. Porovnáme pak i pravděpodobnost, že nalezneme minimální řez. Uvidíme, že tato drobná úprava má za následek obrovské zlepšení.

Věta 1.12. *Algoritmus MinCut je složitosti $O(n^2)$ při libovolném grafu o n vrcholech.*

Důkaz. Uvažme, že každý multigraf si lze představit jako prostý graf, ve kterém může být mezi dvěma libovolnými vrcholy nanejvýš jedna hrana, avšak každé hraně přiřadíme dle její násobnosti váhu. Velikost řezu pak představuje suma vah přeříznutých hran. V tomto případě pak celý graf obsahuje nanejvýš $\binom{n}{2}$ hran. Po přiblížení $\frac{n\cdot(n-1)}{2} = O(n^2)$ je již tvrzení zřejmé. □

Věta 1.13. *Algoritmus FastMinCut je složitosti $O(n^2 \ln n)$ při libovolném grafu o n vrcholech.*

Důkaz. Hloubka rekurze je v řádu $O(\ln n)$, neboť když zkoumáme zmenšování počtu vrcholů z původního grafu o n vrcholech a položíme q jako samotnou hloubku, zkoumáme:

$$6 \geq \frac{n}{(\sqrt{2})^q} \geq 1,$$

jehož jednoduchou logaritmickou úpravou dojdeme k výsledku, že q je řádově rovno $q = O(\ln n)$.

Z předchozí věty tedy víme, že z grafu o n vrcholech vytvoříme graf o dvou vrcholech řádově v čase $O(n^2)$, proto z grafu H uděláme dva grafy H_1 a H_2 v čase $O(n^2)$. Označíme-li tedy jako čas běhu algoritmu na grafu o n vrcholech jako $T(n)$, dostaneme následující rekurzi:

$$T(n) = 2T\left(\left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil \right) + O(n^2).$$

Jestliže neuvažujeme přesné řešení, ale jen řádové, tak můžeme rovnici zjednodušit na

$$T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + O(n^2).$$

Řešení této rovnice nám poskytne tzv. *mistrovská věta* (v angličtině *Master Theorem*) [18] pro asymptotické řešení rekuretní složitosti algoritmů, odkud po dosazení našich hodnot získáme řešení $T(n) = O(n^2 \ln n)$.

□

Nyní bychom ještě chtěli zjistit, o kolik je tento vylepšený algoritmus přesnější neboli jestli dosáhneme lepší pravděpodobnosti, že nalezený řez je tím minimálním. Než vyslovíme a dokážeme danou větu, uveďme si nejprve pomocné lemma, které pak využijeme.

Lemma 1.14. *Mějme graf o n vrcholech s minimálním řezem velikosti k. Označme jev X_m, $m \leq n$ značící, že hrany z minimálního řezu nebudou zkráceny algortimem MinCut do chvíle, kdy graf tvoří právě m hran. Pak platí:*

$$P(X_m) \geq \frac{\binom{m}{2}}{\binom{n}{2}} = \Omega\left(\left(\frac{m}{n}\right)^2\right).$$

Důkaz. Větu 1.10 můžeme přeformulovat jako

$$P(X_2) \geq \prod_{i=1}^{n-2} \Big(1 - \frac{2}{n-i+1}\Big) = \frac{1}{\binom{n}{2}}.$$

Když uvážíme, že jsme ve větě 1.10 zkracovali $n-2$ hran, tak v našem lemmatu zkracujeme jen $n-m$ hran. Rozepíšeme-li daný produkt a rozšíříme-li pak vztah vhodnou jedničkou, dostaneme přímo tvrzení:

$$P(X_m) \geq \prod_{i=1}^{n-m} \Big(\frac{n-i-1}{n-i+1}\Big) = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \cdots \cdot \frac{m}{m+2} \cdot \frac{m-1}{m+1} = \frac{\binom{m}{2}}{\binom{n}{2}}.$$

□

Věta 1.15. *Mějme graf o n vrcholech s minimálním řezem velikosti k. Označme jev A_n, že algoritmus FastMinCut nalezne minimální řez grafu o n vrcholech. Pak $P(A_n) = \Omega\big(\frac{1}{\ln n}\big)$.*

Důkaz. Hlavní myšlenka důkazu je převzata z [7]. Náš algoritmus funguje na bázi rekurze, kdy začínáme na n vrcholech a v každé fázi z grafu o t vrcholech vytvoříme *MinCutem* dva grafy o $t_0 := \left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil$ vrcholech. Jako obvykle bude opět snazší vyjádřit pravděpodobnost komplementárního jevu neboli za využití (1.2) budeme počítat

$$P(A_t) = 1 - P(A_t^C). \tag{1.5}$$

Pravděpodobnost, že nenalezneme minimální řez grafu o t vrcholech, znamená, že ji nenalezneme ani v jednom z obou chodů algoritmu. Jelikož jsou ale jevy nezávislé a zároveň algoritmus poběží v obou chodech na stejném principu, jsou tyto pravděpodobnosti identické. Po označení jevu B_t, že najdeme *min-cut* grafu o t vrcholech během jednoho běhu algoritmu, můžeme psát

$$P(A_t^C) = P(B_t^C)^2.$$

Nyní však opět využijeme (1.2) a píšeme

$$P(A_t^C) = \big(1 - P(B_t)\big)^2.$$

Jestliže ale algoritmus zkracuje hrany grafu o t vrcholech do doby, než vytvoří graf o t_0 vrcholech, a na něj pak rekurzivně opět pouští stejný algoritmus, můžeme použít lemma 1.14 společně s jejím značením a dostaneme

$$P(A_t^C) = \big(1 - P(X_{t_0})P(A_{t_0})\big)^2.$$

Dále dosadíme do Lemma 1.14 a upravíme

$$P(X_{t_0}) \geq \frac{\binom{t_0}{2}}{\binom{t}{2}} = \frac{\left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil \left(\left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil - 1 \right)}{t(t-1)} \geq \frac{\left(1 + \frac{t}{\sqrt{2}}\right)\left(1 + \frac{t}{\sqrt{2}} - 1\right)}{t(t-1)} = \frac{t(\sqrt{2}+t)}{2t(t-1)} \geq \frac{1}{2}.$$

Poskládáme-li tedy dílčí výsledky dohromady, z (1.5) pak vzejde rekurzivní nerovnost

$$P(A_t) \geq 1 - \Big(1 - \frac{1}{2}P(A_{t_0})\Big)^2, \quad t_0 := \left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil.$$

Pro snazší zápis si ještě označíme $P(t) := P(A_t)$.

Nakonec budeme-li řešit rovnici opět jen řádově, postačíme si s přiblížením rovnice

$$P(t) \geq 1 - \left(1 - \frac{1}{2}P\left(\frac{t}{\sqrt{2}}\right)\right)^2.$$

Abychom mohli řešit rovnici pomocí obvyklých metod pro rekurentní rovnice, provedeme transformaci proměnných. Z úvahy o binárním stromu a důkazu věty 1.13 totiž víme, že hloubka stromu grafu o t vrcholech („počet pater“) je řádově $j = O(\ln t)$. S využitím tohoto poznatku bychom mohli od proměnné počtu vrcholů přejít k nové proměnné – hloubce rekurze („hloubky stromu“) a zadefinujeme

$$p(k) = P\left(t\left(\frac{1}{\sqrt{2}}\right)^{j-k}\right) = P\left(t \cdot 2^{\frac{k-j}{2}}\right) \quad \forall k \in \{0, \ldots, j\}.$$

Potom můžeme rovnice ekvivalentně přepsat

$$P(t) \geq 1 - \left(1 - \frac{1}{2}P\left(\frac{t}{\sqrt{2}}\right)\right)^2 \iff p(k) \geq 1 - \left(1 - \frac{1}{2}p(k-1)\right)^2$$

a nahrazením nerovnosti rovností řešíme stejně jako v [19]

$$p(k) = 1 - \left(1 - \frac{1}{2}p(k-1)\right)^2 = p(k-1) - \frac{p(k-1)^2}{4}, \quad p(0) = 1.$$

Vidíme ale, že posloupnost $p(k)$ klesá a konverguje k nule. Obvykle je snazší pracovat naopak s posloupnostmi rostoucími, proto naposledy definujme $q(k) = \frac{1}{p(k)}$. Dosazením pak dostáváme vztah

$$q(k) = \frac{4q(k-1)^2}{4q(k-1)-1} \cdot \frac{4}{4} = \frac{16q(k-1)^2 - 4q(k-1) + 4q(k-1)}{16q(k-1)-4} = q(k-1) + \frac{4q(k-1)-1+1}{16q(k-1)-4}$$

neboli

$$q(k) = q(k-1) + \frac{1}{4} + \frac{1}{16q(k-1)-4}, \quad q(0) = 1.$$

Jelikož víme, že pro každé $k \in \mathbb{N}$ platí $q(k) \geq 1$, můžeme provést následující dva odhady:

- $\frac{1}{16q(k-1)-4} \geq 0$, proto $q(k) \geq q(k-1) + \frac{1}{4}$, odkud se dá snadno vyčíst, že

$$q(k) \geq 1 + \frac{k}{4},$$

- $\frac{1}{16q(k-1)-4} \leq \frac{1}{4}$, proto $q(k) \leq q(k-1) + \frac{1}{2}$, z čehož plyne, že

$$q(k) \leq 1 + \frac{k}{2}.$$

Vidíme tedy, že $q(k) = \Omega(k)$, odkud dostáváme $p(k) = \Omega(\frac{1}{k})$ neboli $P(t) = \Omega(\frac{1}{\ln t})$. Proto $P(A_n) = \Omega\big(\frac{1}{\ln n}\big)$.

□

Jak jsme tedy uvedli dříve, samotný *MinCut* byl složitosti $O(n^2)$ a *min-cut* našel s pravděpodobností $\Omega(\frac{1}{n^2})$. Nyní jsme však sestrojili algoritmus *FastMinCut*, který je sice složitostí větší, resp. $O(n^2 \ln n)$, avšak dosahuje daleko větší úspěšnosti $\Omega(\frac{1}{\ln n})$. Stojí ovšem za zmínku, že byť naší hlavní argumentací nízké úspěšnosti algoritmu *MinCut* byl možný dlouhodobý přenos chyby z prvních několika zkrácení, tak algoritmu *FastMinCut* se to může stát úplně stejně. Rekurzivím dvojnásobným puštěním náhodného algoritmu na stejný graf však znatelně snižujeme pravděpodobnost, že v obou krocích zkrátíme opravdu hranu minimálního řezu.

1.6.3 Kód

Nakonec si ještě uveďme možný kód, pomocí kterého lze algoritmus implementovat. Jedná se jen o variaci na algoritmus *MinCut*.

```
import random

# Function removing self loops
def remove_self_loops(graph):
    for node in graph:
        graph[node] = [neighbor for neighbor in graph[node] if
            neighbor != node]

def min_cut(graph,decision):
    """
    The input is a dictionary that represents the graph in the
       following format: {node1: [neighbor1, neighbor2, ...], node2:
       [neighbor1, neighbor2, ...], ...}
    """
    #Decide, if we make whole min_cut or only partial min_cut
    m=0
    remove_self_loops(graph)
    if decision == 0:
        # Contract edges till 2 vertices remain
        m = 2
    else:
        # Contract edges of a small subset of vertices
        m = int(len(list(graph.keys())) / 2**(1/2))

    while len(graph) > m:
        # Choose a random edge
        node1 = random.choice(list(graph.keys()))
        node2 = random.choice(graph[node1])
        # Contract it
        for w in graph[node2]:
            # Remove edges that are the same as the contracted one
            if w == node1:
                graph[w].remove(node2)
            # Else put all other edges of node2 to node1 and set new
               neighbors of all other vertices w
            else:
                graph[node1].append(w)
                graph[w].remove(node2)
                graph[w].append(node1)
        del graph[node2]

    # Return the contracted graph
    return graph
```

```
40
41 def fast_min_cut(graph,cuts):
42     """
43     Saves the minimum cut of a graph using double recursion and a
          merge fraction of 1/ 2**(1/2). The input graph should be
          represented as a dictionary too. The second input is an array
          of found cuts.
44     """
45
46     # Find number of vertices and remove self loops
47     vertices = list(graph.keys())
48     n = len(vertices)
49     remove_self_loops(graph)
50
51     # Base case: if the graph has from 6 to 2 vertices, append the
          cut using min_cut
52     if 6>=n>=2:
53         graph0 = min_cut(graph,0)
54         node1 = list(graph0.keys())[0]
55         cut = len(graph0[node1])
56         cuts.append(cut)
57     # Recursive case: compute recursively the minimum cut using twice
          the partial_min_cut
58     else:
59         # Contract the graph by merging a small subset of vertices
60         graph1 = min_cut(graph.copy(),1)
61         graph2 = min_cut(graph.copy(),1)
62         fast_min_cut(graph1,cuts)
63         fast_min_cut(graph2,cuts)
64
65 def find_min_cut(graph):
66     # Save found min-cuts into list "cuts" and then choose the
          minimal
67     global cuts
68     cuts=[]
69     fast_min_cut(graph,cuts)
70     #print("cuts:", cuts)
71     print("Min-cut nalezeny algoritmem FastMinCut je:",min(cuts))
```

Kapitola 2

Disperze

V následující kapitole se budeme zabývat problematikou disperze. Nejedná se vůbec o úlohu se složitou implementací algoritmu, o to však náročnější je samotná matematická analýza. Jde o tak složitý problém, že jej nelze vyřešit přímo a přesně, ale dokážeme ho jen pouze odhadnout a omezit shora a zdola. Cílem matematiků po celém světě je pak tento rozdíl v odchadech ze dvou stran učinit co nejmenším a tím tak získat co nejlepší představu o chování veličiny. V textu níže uvedeme některé ze známých odhadů a představíme také vylepšení či alespoň novou alternativu odhadu.

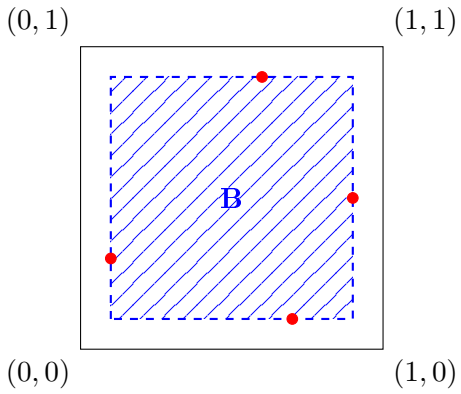
Úmluva 2.1. Nechť $d \in \mathbb{N}$. Zobecněním krychle ve 3 dimenzích do d dimenzí rozumíme d-krychli, zkráceně pouze krychli. Analogicky definujeme d-kvádr, resp. kvádr.

Mějme tedy v prostoru $\mathbb{R}^d, d \in \mathbb{N}$ krychli $[0,1]^d$ a v ní libovolnou množinu bodů $\mathcal{P} \subset [0,1]^d$. Chtěli bychom zkoumat, jaký je největší možný objem kvádru v dané krychli, který neobsahuje ani jeden bod z množiny $\mathcal{P}$ a zároveň má osy rovnoběžné s krychlí.

Definice 2.2. Nechť $\mathcal{P} \subset [0,1]^d, d \in \mathbb{N}$ je množina bodů prostoru $\mathbb{R}^d$. Disperzí množiny $\mathcal{P}$ rozumíme

$$\mathrm{disp}(\mathcal{P}) := \sup_{B:\, B \cap \mathcal{P} = \emptyset} |B|,$$

kde $B = I_1 \times \cdots \times I_d$, $\forall j \in \hat{d}$: $I_j \subset [0,1]$ je kvádr s osami rovnoběžnými s krychlí a symbolem $|B|$ myslíme jeho objem $|B| = \prod_{j=1}^{d} |I_j|$.



Obrázek 2.1: Kvádr B tvořící disperzi $\mathcal{P} = \{(0.7, 0.1), (0.9, 0.5), (0.1, 0.3), (0.6, 0.9)\}$ v $\mathbb{R}^2$.

Nás však bude zajímat takové rozložení bodů v krychli, že bude disperze co nejmenší. Ilustrovat by se to dalo ve dvou dimenzích na jehličkových tiskárnách v dobách, kdy se tisklo pouze černobíle. Stínování obrázku se tvořilo pomocí hustoty černých teček na papíře – čím více puntíků, tím více tmavší barva. Kvůli úsporám barvy se ale hledalo takové uspořádání teček, aby spotřeba inkoustu byla co nejmenší, ale zároveň kvalita obrázku co nejvyšší. Praktické aplikace pak úloha nachází v data miningu při hledání co největších prázdných oblastí v datech [20] nebo při vystřihování železných kvádrů z krychlí, přičemž se chceme vyhnout poškozeným částem, a jinde [12].

Definice 2.3. Nechť $n, d \in \mathbb{N}$. Pak n-tou minimální disperzi krychle $[0,1]^d$ definujeme jako

$$\operatorname{disp}(n,d) := \inf_{\substack{\mathcal{P}\subset[0,1]^d \\ \#\mathcal{P}=n}} \operatorname{disp}(\mathcal{P})$$

a její inverzní funkci

$$N(\varepsilon,d) := \min\{n\colon \operatorname{disp}(n,d) \le \varepsilon\}. \tag{2.1}$$

Přestože to na první pohled není zřejmé, jedná se o složitý problém mnohaletého bádání matematiků. Není těžké si rozmyslet, že pro každé přirozené d bude $\operatorname{disp}(n,d)$ přibližně n^{-1} pro $n \to \infty$. V sekci 2.3 uvedeme aktuálně nejlepší odhady na disperzi jak shora, tak zdola. Abychom však provedli odhad zespod, tak si za tímto účelem uvedeme odhad shora, který dokázal Mario Ullrich a Jan Vybíral [21], neboť využijeme poznatky a princip z tohoto důkazu. Do té doby, než byl představen tento odhad [21], byly nejlepší známé odhady

$$\frac{\log_2(d)}{4(n+\log_2(d))} \le \operatorname{disp}(n,d) \le \frac{C^d}{n} \quad \forall n,d \in \mathbb{N},\ C \in \mathbb{R}^+, \tag{2.2}$$

kde dolní odhad dokázal Christoph Aistleitner a spol. [22] a horní odhad získal Gérald Larcher a dokázal opět Christoph Aistleitner a spol. [22, Kapitola 4]. Zdá se, že závislost disperze na dimenzi a počtu bodů by mohla být $\operatorname{disp}(n,d) \approx \frac{\ln d}{n}$, avšak prozatím ještě nebyla nikým dokázaná pro každé $n, d \in \mathbb{N}$. Aktuální srovnání uvedených odhadů shora a zespod s těmi doposud nejlepšími uvedeme v sekci 2.3.

2.1 Horní odhad

Budeme-li nyní zkoumat zmíněnou invezní funkci $N(\varepsilon,d)$ z definice 2.3, Mario Ullrich a Jan Vybíral [21] dokázali důsledek 2.12, jenž omezuje disperzi shora. Avšak pomocí jejich důkazu se lze dopracovat k lepšímu výsledkun než je uveden v daném článku. Uveďme si tedy větu 2.6, z níž pak důsledek 2.12 již vyplývá. Než ji však představíme a nastíníme důkaz, uvažujme následující definici. Myšlenka důkazu je totiž v přenesení problému ze spojitého do diskrétního případu. Sestrojíme si tedy d-dimenzionální mřížku, ze které budeme volit n bodů. Vezmeme pak všechny kvádry, které mají objem větší než 2^{-k} pro jisté $k \in \mathbb{N}$ a omezíme nějakým způsobem jejich souřadnice. Když pak pro každý kvádr bude existovat nějaký bod, který kvádr protne, bude to znamenat, že disperze je menší než 2^{-k}. Diskretizace je tedy následující.

Definice 2.4. Nechť $k \in \mathbb{N}$. Pak definujeme

$$\Omega_k := \{B \subset [0,1]^d\colon\ B \text{ je kvádr s rovnoběžnými osami} \wedge |B| > 2^{-k}\}, \tag{2.3}$$

respektive

$$M_k := \left\{ \frac{1}{2^k}, \frac{2}{2^k}, \ldots, \frac{2^k-1}{2^k} \right\} \subset [0,1].$$

Sestrojíme-li pak vektory $p = (p_i)_{i=1}^d \in M_k^d$ a $s = (s_i)_{i=1}^d \in \{0,1,\ldots,2^k-1\}^d$, definujeme množinu

$$\Omega_k(p,s) := \left\{ I_1 \times \cdots \times I_d \in \Omega_k \colon \ \frac{s_l}{2^k} < |I_l| \leq \frac{s_l+1}{2^k} \wedge \ \inf I_l \in \left[p_l - \frac{1}{2^k}, p_l \right) \ \forall l \in \hat{d} \right\}. \quad (2.4)$$

Poznámka 2.5. V následujícím textu uvažujme, že všechny hrany kvádrů $I_1, \ldots, I_d$ jsou otevřenými intervaly. Velikost disperze definovaná pomocí suprema bude pak stejné hodnoty jako pro uzavřené intervaly.

Věta 2.6. *Nechť $d \geq 2$, $d \in \mathbb{N}$ a $\varepsilon \in (0, \frac{1}{2})$. Pak existuje množina bodů $\mathcal{P} \subset [0,1]^d$ s disperzí $disp(\mathcal{P}) \leq \varepsilon$ a zároveň*

$$\#\mathcal{P} \leq 2^6 \frac{1}{\varepsilon^2} \left(1 + \log_2\left(\frac{1}{\varepsilon}\right)\right)\left(2 + \log_2\left(\frac{1}{\varepsilon}\right) + \log_2(d)\right). \quad (2.5)$$

Důkaz. Nebudeme zde uvádět kompletní důkaz, ten je ostatně k nahlédnutí v samotném článku [21]. Pokusíme se však nastínit alespoň jeho hlavní myšlenku a princip.

Potřebujeme dokázat, že existuje taková množina bodů $\mathcal{P}$ s požadovanou velikostí, jejíž disperze je omezená ε. To znamená, že každý kvádr o objemu ε už obsahuje alespoň jeden bod z $\mathcal{P}$. Postupujme tedy tak, že zkonstruujeme testovací množinu všech kvádrů o objemu minimálně ε, které obsahuje daná krychle $[0,1]^d$. Následně budeme náhodně volit body $X = \{x^1, \ldots, x^n\}$ z krychle a snažit se vyjádřit pravděpodobnost, že existuje alespoň jeden bod $x^i, i \in \hat{n}$, který nějaký kvádr protne. Pokud tato pravděpodobnost bude nenulová, bude to dokazovat existenci takové množiny bodů, která protne každý kvádr o objemu větším než ε, a tím bude důkaz hotov.

Jelikož možných kvádrů, které může krychle obsahovat, je nekonečně mnoho, musíme ze spojitého případu přejít nějakým způsobem do spočetné diskrétní roviny. Avšak stále nesmíme zapomenout na to, že kvádrů je nekonečně mnoho. Tento krok provedeme následovně. Za účelem využití definice 2.4 sestrojíme takové $k \in \mathbb{N}$, abychom omezili daný objem kvádrů o objemu alespoň $0 < \varepsilon < \frac{1}{2}$ tak, že

$$2^{-k} \leq \varepsilon < 2^{-k+1}. \quad (2.6)$$

Neboli k je alespoň

$$k = \left\lceil \log_2\left(\frac{1}{\varepsilon}\right) \right\rceil \geq 2.$$

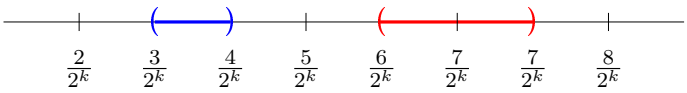
Tímto k pak definujeme množinu M_k a množinu kvádrů $\Omega_k(p,s)$. Pro zadané p a s dostáváme konkrétní $\Omega_k(p,s)$, pro které platí, že všechny $B \in \Omega_k(p,s)$ mají skoro stejné souřadnice. O tom pojednává následující lemma.

Lemma 2.7. *Nechť $M_k^d(p,s) = \bigtimes_{l=1}^{d} M_k^l(p_l, s_l)$, kde $\forall l \in \hat{d}$: $M_k^l(p_l, s_l) = \{p_l, \ldots, p_l + \frac{s_l-1}{2^k}\}$. Pak pro všechna $B \in \Omega_k(p,s)$ platí:*

$$B \cap M_k^d \supset M_k^d(p,s).$$

Důkaz. Nejvíce by se nám hodilo, kdyby všechny průniky kvádrů z $\Omega_k(p,s)$ s M_k^d byly stejné. To však při naší konstrukci nejde, jelikož z vlastností $\Omega_k(p,s)$ plyne, že

$$\inf I_l \in \left[p_l - \frac{1}{2^k}, p_l\right) \quad \wedge \quad \sup I_l \in \underbrace{\left(p_l + \frac{s_l - 1}{2^k}, p_l + \frac{s_l + 1}{2^k}\right)}_{|\cdot| = \frac{2}{2^k}} \quad \forall l \in \hat{d}.$$

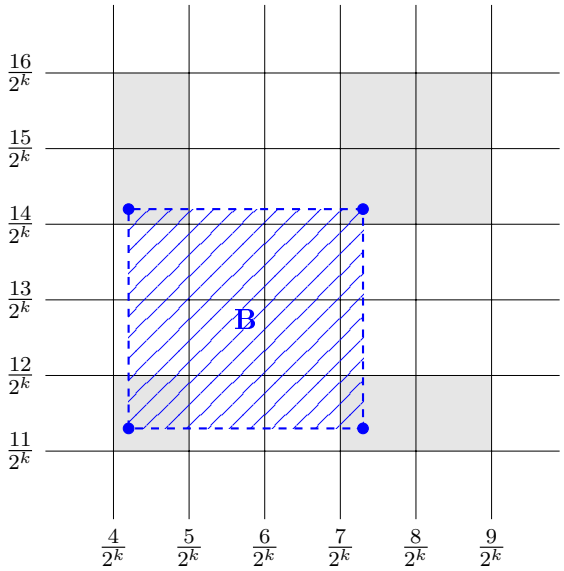


Obrázek 2.2: Znázornění, kde všude se může nacházet $\inf I_l$ (modře), respektive $\sup I_l$ (červeně) pro $p_l = \frac{4}{2^k}$ a $s_l = 3$.

Proto může $\sup I_l$ padnout do intervalu délky $\frac{2}{2^k}$ a průniky se tedy mohou pro různá $B \in \Omega_k(p,s)$ lišit o jednu souřadnici. Pokud ale vezmeme největší společný průnik, počínaje p_l a končeje $p_l + \frac{s_l - 1}{2^k}$, dojdeme k tvrzení lemma, že

$$\forall l \in \hat{d}: \quad I_l \cap M_k \supset \left\{p_l, \ldots, p_l + \frac{s_l - 1}{2^k}\right\} =: M_k^l(p_l, s_l).$$

□



Obrázek 2.3: Kvádr B v $\mathbb{R}^2$, kde $s = (3,3)$ a $p = \left(\frac{5}{2^k}, \frac{12}{2^k}\right)$. Šedě jsou znázorněna místa, kde se mohou nacházet vrcholy všech kvádrů v $\Omega_k\left(\left(\frac{5}{2^k}, \frac{12}{2^k}\right), (3,3)\right)$.

Pro různá p a s pak dostáváme různé průniky $M_k^d(p,s)$. Těmi pak pokryjeme celý prostor krychle. Pokud následně zvolíme nějaký bod z $x \in M_k^d$ a trefí se do $M_k^d(p,s)$, trefí se zároveň do všech $B \in \Omega_k(p,s)$. Tím máme pro různá p a s pokryty všechny kvádry z $[0,1]^d$. Naším snažením pak bude mít co nejméně p a s, ale zároveň co největší $M_k^d(p,s)$. To jsou však protichůdné požadavky a my se budeme snažit najít jistou rovnováhu.

Uvažujme tedy náhodně zvolenou množinu bodů $X = \{x^1, \dots, x^n\} \subset M_k^d$ velikosti n, kde souřadnice $(x^j)_l$, $j \in \hat{n}$ a $l \in \hat{d}$ jsou náhohodně a uniformně vybrané. Důležité v této chvíli je si uvědomit, že $\Omega_k(p,s)$ je rozdělení Ω_k a že pro některé volby p a s je $\Omega_k(p,s) = \emptyset$. Například pokud existuje index $i \in \hat{d}$ takový, že $s_i = 0$, je pak $|I_i| \leq \frac{1}{2^k}$ a pro takové B pak platí $|B| \leq \frac{1}{2^k}$, tudíž již nemůže patřit do Ω_k, a tedy ani do $\Omega_k(p,s)$.

Následně se v důkazu využije [21, Lemma 3]. Toto lemma nám v podstatě říká, že naše $M_k^d(p,s)$ jsou opravdu velké (připomínáme, že objem každého $B \in \Omega_k(p,s)$ je větší než 2^{-k}):

Lemma 2.8. *Nechť x je vybráno uniformně z M_k^d. Pak pro každé $p \in M_k^d$ a $s \in \{0, \dots, 2^k - 1\}^d$ taková, že $\Omega_k(p,s) \neq \emptyset$, platí:*

$$P(\forall B \in \Omega_k(p,s) \colon x \in B) \geq P(x \in M_k^d(p,s)) > \frac{1}{2^{k+4}} \tag{2.7}$$

neboli

$$P(\exists B \in \Omega_k(p,s) \colon x \notin B) < e^{-\frac{1}{2^{k+4}}}. \tag{2.8}$$

Důkaz. Ponecháme bez důkazu (viz [21]). Uvedeme jen, že důkaz v podstatě využívá samotné lemma 2.7. □

Vlastní důkaz tedy spočívá v hledání pravděpodobnosti, že zvolená množina bodů neprotne nějaký kvádr z Ω_k, tedy

$$\begin{aligned} P(\exists B \in \Omega_k \colon\ X \cap B = \emptyset) &\leq \sum_{p,s \colon \Omega_k(p,s) \neq \emptyset} P(\exists B \in \Omega_k(p,s) \colon X \cap B = \emptyset) \\ &= \sum_{p,s \colon \Omega_k(p,s) \neq \emptyset} P(\exists B \in \Omega_k(p,s) \colon x_1 \notin B)^n \\ &\overset{\text{L. 2.8}}{<} \#\{(p,s) \colon \Omega_k(p,s) \neq \emptyset\} \cdot e^{-\frac{n}{2^{k+4}}}. \end{aligned}$$

Nyní se důkaz přenese do hledání počtu (p,s) takových, že $\Omega_k(p,s) \neq \emptyset$. Z úvahy výše, že existují volby p a s takové, že $\Omega_k(p,s)$ je prázdná, musíme vznést nějaká omezení na s_i a p_i z definice $\Omega_k(p,s)$, kde $p = (p_i)_{i=1}^d$ a $s = (s_i)_{i=1}^d$. Úloha pak v podstatě přejde ve zkoumání a hledání tzv. pasivních a aktivních souřadnic kvádrů.

Definice 2.9. Nechť $B \in \Omega_k(p,s)$. Řekneme, že $j \in \hat{d}$ je aktivní, resp. pasivní souřadnice kvádru B, jestliže $s_j < 2^k - 1$, resp $s_j = 2^k - 1$.

Poznámka 2.10. Jestliže $j \in \hat{d}$ je aktivní souřadnice, tak pro hranu I_j platí, že $|I_j| \leq \frac{2^k-1}{2^k}$. Neboli hranu I_j můžeme v krychli posunout například tak, že $(0, |I_j|) \subset \left(0, \frac{2^k-1}{2^k}\right) =: I_0$ a tudíž existuje $x \in M_k^d$ takové, že $x \notin I_0$. Naopak pasivní souřadnice znamená to, že ať hranu „umístíme“ kamkoli v krychli, vždy bude libovolný bod $x \in M_k^d$ v této hraně obsažen.

Proto se v důkazu vyskytne $m_1(s) := \#\{i \colon\ s_i < 2^k - 1\}$ jako počet aktivních souřadnic. Hrana I_i je pak totiž dlouhá nanejvýš stejně jako rozdíl krajních hodnot z M_k^d. Avšak z poznámky výše a faktu, že hrana je otevřený interval, již víme, že ji můžeme i tak posunout a nějaký bod se do ní nemusí strefit.

Věta 2.11. *Nechť $B \in \Omega_k(p,s)$ a označme počet aktivních souřadnic kvádru B jako $m_1(s) := \#\{i \colon\ s_i < 2^k - 1\}$. Pak platí*

$$m_1(s) < \ln(2) k 2^k.$$

Důkaz. Z (2.3) víme, že $2^{-k} < |B|$. Zároveň z vlastnosti (2.4) dostaneme, že pro $B = I_1 \times \cdots \times I_d$ platí $|I_l| \leq \frac{s_l+1}{2^k}$ pro všechna $l \in \hat{d}$, z čehož plyne

$$2^{-k} < |B| \leq \prod_{l=1}^{d} \Big(\frac{s_l+1}{2^k}\Big) \leq \Big(1 - \frac{1}{2^k}\Big)^{m_1(s)}.$$

Zlogaritmováním tohoto výrazu získáváme

$$m_1(s) < \frac{\ln(2^{-k})}{\ln(1-2^{-k})} = \frac{k\ln(2)}{-\ln(1-2^{-k})},$$

přičemž po použití nerovnosti

$$-\ln(1+x) \geq -x, \;\; \forall x \in (-1, +\infty) \tag{2.9}$$

dostáváme

$$m_1(s) < k2^k \ln(2).$$

□

Náročnou technickou pasáž zbytku důkazu přenecháme na samotném čtenáři, k náhlednutí je v [21]. Nakonec dojdeme k výsledku

$$P(\exists B \in \Omega_k : \; X \cap B = \emptyset) < e^{k2^k \log_2(2^{k+1}d) - \frac{n}{2^{k+4}}}. \tag{2.10}$$

My však chceme ukázat, že

$$0 < P(\forall B \in \Omega_k : \; X \cap B \neq \emptyset) = 1 - P(\exists B \in \Omega_k : \; X \cap B = \emptyset), \tag{2.11}$$

abychom tím potvrdili existenci takové množiny X. Když však výraz napravo v (2.10) omezíme jedničkou, bude tím (2.11) dokázána. Pokud tak uděláme, dojdeme k výsledku, že

$$e^{k2^k \log_2(2^{k+1}d) - \frac{n}{2^{k+4}}} \leq 1 \iff n \geq 2^4 k 2^{2k} \log_2(2^{k+1}d). \tag{2.12}$$

Neboli vezmeme-li takové n, tak existuje množina bodů $\{x^1, \ldots, x^n\} \subset M_k^d$ taková, že skutečně prostřelí každý kvádr B o objemu $|B| > 2^{-k}$. To přímo implikuje skutečnost, že $\mathrm{disp}(n,d) \leq \varepsilon = 2^{-k}$. Jelikož ale $N(\varepsilon, d)$ je infimum přes všechna n taková, že $\mathrm{disp}(n,d) \leq \varepsilon$, bude proto nanejvýš stejně velké jako (2.12). Proto tedy musí platit

$$N(\varepsilon, d) \leq 2^4 k 2^{2k} \log_2(2^{k+1}d).$$

Dosaďme za k ze vztahu (2.6)

$$k \leq 1 + \log_2\Big(\frac{1}{\varepsilon}\Big),$$

$$2^k \leq 2\frac{1}{\varepsilon},$$

a dostaneme tvrzení samotné věty

$$N(\varepsilon, d) \leq 2^6 \frac{1}{\varepsilon^2}\Big(1 + \log_2\Big(\frac{1}{\varepsilon}\Big)\Big)\Big(2 + \log_2\Big(\frac{1}{\varepsilon}\Big) + \log_2(d)\Big).$$

□

Důsledek 2.12. *Nechť $d \geq 2$, $d \in \mathbb{N}$ a $\varepsilon \in (0, \frac{1}{2})$. Pak existuje množina bodů $\mathcal{P} \subset [0,1]^d$ s disperzí $disp(\mathcal{P}) \leq \varepsilon$ a zároveň*

$$\#\mathcal{P} \leq 2^7 \log_2(d) \frac{\left(1 + \log_2(\frac{1}{\varepsilon})\right)^2}{\varepsilon^2}. \tag{2.13}$$

Důkaz. Pro všechna $a \geq 2$ a $b \geq 1$ platí $1 + a + b \leq 2ba$. To lze jednoduše dokázat

$$1 + a + b \leq 2ba = ba + ba \iff 1 \leq b\underbrace{(a-1)}_{\geq 1} + a\underbrace{(b-1)}_{\geq 0}.$$

Pokud pak do této nerovnosti dosadíme z věty 2.6 za $a = 1 + \log_2\left(\frac{1}{\varepsilon}\right)$ a $b = \log_2(d)$, je pak důsledek zřejmý. □

2.2 Dolní odhad

Náš odhad na $N(\varepsilon, d)$ zkusme ještě omezit z druhé strany. K tomu nám poslouží jiný matematický problém, tentokráte bližší kombinatorické matematice, na který pak dokážeme přetransformovat problém disperze. Ten spočívá v odhadování toho, kolik vektorů, které se skládají pouze z nul a jedniček, musí být, aby na jistých pozicích byly jedničky a na jiných zase nuly. Korektní definice takzvaného (l,d)-restriction setu je pak následující.

Úmluva 2.13. Nechť $N, d \in \mathbb{N}$, $x^1, \ldots, x^N \in \{0,1\}^d$ a $A \subset \{1, \ldots, d\}$. Poté pro $u \in \hat{N}$: $(x^u)\big|_A = 1$ rozumíme, že pro každé $i \in A$ platí $(x^u)_i = 1$.

Definice 2.14. Nechť $N, l, d \in \mathbb{N}$ taková, že $1 \leq l \leq d$. Řekneme, že množina bodů $x^1, \ldots, x^N \in \{0,1\}^d$ je (l,d)-restriction set, jestliže $\forall A \subset \{1, \ldots, d\}$: $|A| = l - 1$ a $\forall j \in \{1, \ldots, d\} \setminus A, \exists u \in \hat{N}$: $(x^u)_j = 0 \wedge (x^u)\big|_A = 1$. Následně pak definujme velikost nejmenšího (l,d)-restriction setu jako

$$R(l,d) = \min\{N \in \mathbb{N}\colon \exists\{x^1, \ldots, x^N\} \subset \{0,1\}^d, \text{který je } (l,d)\text{-restriction set}\}.$$

Poznámka 2.15. V následující pasáži se dostaneme k odhadům na $R(l,d)$. Proto si pro pochopení pojmu uveďme jisté triviální odhady, které lze ihned odvodit:

- $l \leq R(l,d)$: pokud vezmeme pevné $j \in \{1, \ldots, l\}$ a k němu $A = \{1, \ldots, l\} \setminus j$, pak máme l možností, jak dané j volit. Avšak je zřejmé, že tímto přístupem nepokryjeme všechny množiny A o velikosti $|A| = l - 1$, respektive jsme pro každé j uvážili jen jednu takovou, proto musí být $R(l,d)$ větší.
- $R(l,d) \leq \binom{d}{l-1}$: vezmeme všechny vektory, které mají $l - 1$ jedniček a zbytek jsou nuly. Avšak z velikosti kombinačního čísla není odhad vůbec optimální.
- $R(l,d) \leq d$: pokud vezmeme množinu vektorů $\{x^1, \ldots, x^d\}$, pro kterou platí, že vektor x^i má na všech složkách jedničky až na $(x^i)_i = 0$, je množina (l,d)-restriciton set.

Stejně jako v předešlé kapitole, mějme zadané $0 < \varepsilon < \frac{1}{2}$ a zkoumejme disperzi krychle $[0,1]^d$ neboli hledejme $N(\varepsilon, d)$. Stejně jako v (2.6) nalezneme $k \in \mathbb{N}$ takové, že $\varepsilon \approx 2^{-k}$. Poté uvažme následující množinu testovacích kvádrů.

Definice 2.16. Nechť $k, d \in \mathbb{N}$, $A \subset \{1, \ldots, d\}$ a $j \in \{1, \ldots, d\} \setminus A$. Pak $B_{j,A} = I_1 \times \cdots \times I_d \subset [0,1]^d$ je takový kvádr, pro který platí:

- $I_j = \left[0, \frac{2}{2^k}\right)$,
- $\forall i \in A\colon I_i = \left(\frac{2}{2^k}, 1\right]$,
- $\forall l \in \{1, \ldots, d\} \setminus (A \cup j)\colon I_l = [0, 1]$.

Nakonec definujme $\mathcal{B} = \{B_{j,A}\colon A \subsetneq \{1, \ldots, d\}, A \neq \emptyset \ \wedge \ j \in \{1, \ldots, d\} \setminus A\}$.

Opět budeme chtít, aby $\mathcal{B}$ byla množina testovacích kvádrů neboli aby obsahovala kvádry o objemu větším něž 2^{-k}. Proto zkoumejme vztah mezi objemem kvádrů z $\mathcal{B}$ a velikostí A.

Věta 2.17. *Nechť $k \in \mathbb{N}$ a $k \geq 2$. Pak pro všechna $B_{j,A} \in \mathcal{B}$ platí následující dvě implikace:*

1. *Jestliže $|B_{j,A}| > 2^{-k}$, tak $|A| < \frac{\ln 2}{2} 2^k$.*
2. *Jestliže $|A| < \frac{1}{4} 2^k$, pak $|B_{j,A}| > 2^{-k}$.*

Důkaz. 1. Předpokládáme, že pro $B_{j,A} \in \mathcal{B}$ platí $|B_{j,A}| > 2^{-k}$. Z toho plyne

$$|B_{j,A}| > 2^{-k} \iff \frac{2}{2^k}\left(1 - \frac{2}{2^k}\right)^{|A|} > \frac{1}{2^k}$$

a po zlogaritmování výrazu zjistíme, že

$$|A| < \frac{\ln 2}{-\ln\left(1 - \frac{2}{2^k}\right)} \overset{(2.9)}{\leq} \frac{\ln 2}{2} 2^k.$$

2. Máme omezenou velikost množiny A a potřebujeme dokázat, že

$$\left(1 - \frac{2}{2^k}\right)^{|A|} > \frac{1}{2},$$

což je ekvivalentní s otázkou, jestli

$$|A| \ln\left(1 - \frac{2}{2^k}\right) > -\ln 2. \tag{2.14}$$

Vyjdeme-li z předpokladu, že $|A| < \frac{1}{4} 2^k$, a vynásobíme nerovnici $\ln\left(1 - \frac{2}{2^k}\right)$, který je ovšem pro libovolné $k \in \mathbb{N}$ záporný, dostáváme za použití tvrzení 2.18 nerovnici

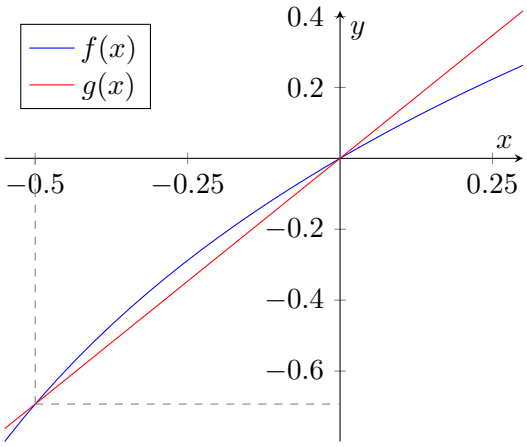
$$|A| \ln\left(1 - \frac{2}{2^k}\right) > \frac{1}{4} 2^k \ln\left(1 - \frac{2}{2^k}\right) \overset{2.18}{\geq} -\ln 2,$$

čímž je nerovnost (2.14) dokázána.

□

Tvrzení 2.18. Pro všechna $x \in \left[-\frac{1}{2}, 0\right]$ platí: $2x \ln 2 \leq \ln(1 + x)$.

Důkaz. Funkce $f(x) = \ln(1 + x)$ je na intervalu $\left[-\frac{1}{2}, 0\right]$ konkávní. Vezmeme-li tedy lineární funkci $g(x)$, která prochází body $f\left(-\frac{1}{2}\right) = -\ln 2$ a $f(0) = 0$, což splňuje funkce $g(x) = 2x \ln 2$, pak je nerovnost zřejmá. □



Obrázek 2.4: Graf funkcí $f(x) = \ln(1+x)$ a $g(x) = 2x \ln 2$ na intervalu $x \in \left[-\frac{1}{2}, 0\right]$.

Nyní již můžeme přistoupit k samotnému odhadu disperze zespod. Jak již bylo nastíněno, pokusíme se to udělat právě pomocí (l,d)-restriction setu. K tomu bychom chtěli co největší množinu A v množině $\mathcal{B}$, nejlépe aby se velikostí blížila k maximálnímu počtu aktivních souřadnic omezenému ve větě 2.11. Jak ale vidíme, pro naši testovací množinu $\mathcal{B}$ se nám to nepodařilo, protože z věty 2.17 můžeme uvažovat A pouze takové, že $|A| \approx 2^k$, tudíž nám chybí ještě logaritmická závislost. Jak si však můžeme všimnout z důkazu věty 2.11, tu získáme, pokud všechny hrany budou délky téměř jedna, respektive $\left(1 - \frac{1}{2^k}\right)$. To by však neprošlo při konstrukci důkazu následující věty 2.19. Museli bychom tedy uvažovat buď jinou množinu, nebo zvolit zcela jiný princip důkazu.

My se však zabývejme pouze tímto konkrétním $\mathcal{B}$. Disperzi pak můžeme zespod omezit pomocí $R(l,d)$ pro jisté l, které budeme chtít mít co největší. O spojení R a N pojednává následující věta, která je velmi zásadní pro další pokračování.

Věta 2.19. *Nechť $k, d \in \mathbb{N}$ jsou taková, že $k \geq 2$ a $2^{k-2} \leq d$. Pak platí*

$$R\big(2^{k-2}, d\big) \leq N\big(2^{-k}, d\big). \tag{2.15}$$

Důkaz. Chceme zespod odhadnout $N(2^{-k}, d)$ neboli minimální počet bodů takových, že disperze této množiny bude menší než 2^{-k}. Položme tedy $N := N(2^{-k}, d)$ a vezměme libovolnou množinu N bodů $\{x^1, \ldots, x^N\} \subset [0,1]^d$.

Využijme nyní množinu kvádrů $\mathcal{B}$ z definice 2.16. Z věty 2.17 však víme, že pro libovolné $B_{j,A} \in \mathcal{B}$, respektive pro každé $j \in \{1, \ldots, d\}$ a libovolné $A \subset \{1, \ldots, d\} \setminus j$: $|A| < \frac{1}{4}2^k$ plyne, že $|B_{j,A}| > 2^{-k}$. Vezmeme-li pak $|A| = \frac{1}{4}2^k - 1$, dostáváme, že též existuje $u \in \hat{N}$ takové, že $x^u \in B_{j,A}$.

Použijeme-li následně zaokrouhlovací funkci

$$\forall t \in [0,1]\colon \phi(t) = \begin{cases} 0, & \text{pokud } t < \frac{2}{2^k}, \\ 1, & \text{pokud } t \geq \frac{2}{2^k}, \end{cases}$$

tak ovšem dostáváme, že pro každé $j \in \{1, \ldots, d\}$ a $A \subset \{1, \ldots, d\} \setminus j$: $|A| = 2^{k-2} - 1$ platí, že existuje $u \in \mathbb{N}$ takové, že $\phi(x^u)_j = 0$ a $\phi(x^u)\big|_A = 1$ (funkci ϕ aplikovanou na vektor definujeme po složkách). Tím jsme však dokázali, že $\{\phi(x^1), \ldots, \phi(x^N)\}$ splňuje $(2^{k-2}, d)$-restriction set a tudíž $R\big(2^{k-2}, d\big) \leq N$, z čehož vyplývá samotné tvrzení. □

Nyní bychom chtěli získat nějaké odhady právě na $R(l,d)$, abychom pak pomocí nich mohli odhadnout minimální disperzi. K tomu nám poslouží následující dvě věty. Ještě předtím si však uveďme odhady na kombinační číslo, které budeme v následujících důkazech potřebovat.

Tvrzení 2.20. Nechť $k, n \in \mathbb{N}$ taková, že $k \leq n$. Pak

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{ne}{k}\right)^k. \tag{2.16}$$

Důkaz. 1. Nejprve dokážeme odhad zespod. Pro $k = 1$ je to triviální nerovnost

$$\left(\frac{n}{1}\right)^1 = n \leq n = \binom{n}{1}.$$

Poté pro $k > 1$ a pro každé $m \in \mathbb{N}$ takové, že $0 < m < k \leq n$ platí, že

$$k \leq n \iff \frac{m}{n} \leq \frac{m}{k} \iff 1 - \frac{m}{k} \leq 1 - \frac{m}{n} \iff \frac{k-m}{k} \leq \frac{n-m}{n} \iff \frac{n}{k} \leq \frac{n-m}{k-m}.$$

Odtud pak dostáváme samotné tvrzení

$$\left(\frac{n}{k}\right)^k = \frac{n}{k} \cdot \ldots \cdot \frac{n}{k} \leq \frac{n}{k} \cdot \frac{n-1}{k-1} \cdot \ldots \cdot \frac{n-k+2}{2} \cdot \frac{n-k+1}{1} = \binom{n}{k}.$$

2. Nyní proveďme odhad shora. Kombinační číslo můžeme přepsat do tvaru

$$\binom{n}{k} = \frac{n(n-1)\ldots(n-k+1)}{k!} \cdot \frac{n^k}{n^k} = \left(1 - \frac{1}{n}\right) \cdots \left(1 - \frac{k-1}{n}\right) \frac{n^k}{k!}.$$

Jelikož každá závorka je ostře menší než jedna, můžeme psát

$$\binom{n}{k} < \frac{n^k}{k!}. \tag{2.17}$$

Pokud však vezmeme Taylorův rozvoj funkce $e^x = \sum_{m=0}^{\infty} \frac{x^m}{m!}$, tak pro libovolné $x > 0$ bude každý člen sumy menší než součet sumy celé neboli

$$\frac{x^m}{m!} < e^x \quad \forall x > 0 \text{ a } m \in \mathbb{N}.$$

Jestliže však zvolíme $x := k$ a $m := k$, pak dostáváme

$$\frac{k^k}{k!} < e^k \iff 1 < \frac{k! e^k}{k^k}. \tag{2.18}$$

Pokud tedy odhad (2.17) ještě zhoršíme pomocí (2.18), dostáváme samotné tvrzení

$$\binom{n}{k} < \frac{n^k}{k!} \cdot \frac{k! e^k}{k^k} = \left(\frac{ne}{k}\right)^k.$$

□

Věta 2.21. *Nechť $l, d \in \mathbb{N}$ jsou takové, že $1 \leq l \leq d$, $d \geq 2$ a $C_1 \in \mathbb{R}^+$ je jistá konstanta. Pak*

$$R(l,d) \leq C_1 l^2 \ln d.$$

Důkaz. Vztah dokážeme pomocí známé pravděpodobnostní metody. Chtěli bychom najít odhad pro N takové, že $\{x^1, \ldots, x^N\}$ je (l,d)-restriction set. Když pak nějaké takové N nalezneme, infimum přes všechna taková N_0, že $\{x^1, \ldots, x^{N_0}\}$ je (l,d)-restriction set, bude muset být nanejvýš stejně velké jako námi nalezené N a tím bude odhad shora hotov. Volme $j \in \{1, \ldots, d\}$ libovolné a $A \subset \{1, \ldots, d\} \setminus j \colon |A| = l-1$. Řekneme, že $x \in \{0,1\}^d$ splňuje (j,A), jestliže $x_j = 0$ a $x\big|_A = 1$. Nechť $x \in \{0,1\}^d$ je náhodně zvolené číslo, kde pro všechna $i \in \hat{d}$ platí: $P(x_i = 1) = p$ a $P(x_i = 0) = 1-p$ pro $p \in (0,1)$. Dále pak budeme postupovat stejně jako v důkazu věty 2.6 – pokud bude pravděpodobnost, že pro libovolné j a A existuje bod $x \in \{x^1, \ldots, x^N\}$, který (j,A) splní, nenulová, pak bude existence celé množiny potvrzena. Postupujme opět přes pravděpodobnost komplementárního jevu (1.2).

Jestliže zvolíme libovolné (j,A), tak pravděpodobnost, že náhodné x splňuje (j,A), je rovna

$$P(x \text{ splňuje } (j,A)) = p^{l-1}(1-p)$$

neboli z (1.2):

$$P(x \text{ nesplňuje } (j,A)) = 1 - p^{l-1}(1-p).$$

Když pak uvážíme množinu $\{x^1, \ldots, x^N\}$ náhodně a nezávisle sestrojenou, pak můžeme psát

$$P(\forall u \in \hat{N} \colon x^u \text{ nesplňuje } (j,A)) = \big(1 - p^{l-1}(1-p)\big)^N.$$

Snadno můžeme zjistit, že $\#\{A \colon |A| = l-1\} = \binom{d}{l-1}$ a k ní můžeme sestrojit $d-(l-1)$ různých j. Pokud následně využijeme vztah

$$\binom{d}{l-1} \cdot (d-l+1) = \frac{d!}{(l-1)!(d-l)!} \cdot \frac{l}{l} = \binom{d}{l} \cdot l,$$

můžeme vyjádřit

$$P(\exists (j,A) \colon \forall u \in \hat{N} \colon x^u \text{ nesplňuje } (j,A)) = P\Big(\bigcup_{(j,A)} \big(\forall u \in \hat{N} \colon x^u \text{ nesplňuje } (j,A)\big)\Big) \leq$$

$$\leq \sum_{(j,A)} \big(1 - p^{l-1}(1-p)\big)^N = \binom{d}{l} \cdot l \big(1 - p^{l-1}(1-p)\big)^N.$$

Využijeme známou nerovnost $1 + x \leq e^x$ pro všechna $x \in \mathbb{R}$ a zároveň odhad (2.16) a dostáváme

$$P(\exists (j,A) \colon \forall u \in \hat{N} \colon x^u \text{ nesplňuje } (j,A)) \leq \Big(\frac{de}{l}\Big)^l \cdot l \cdot e^{-Np^{l-1}(1-p)}.$$

Tuto pravděpodobnost chceme mít co nejmenší, proto nalezneme minimum pro výraz napravo podle p, získanou hodnotu $p = \frac{l-1}{l}$ dosadíme do výrazu a získáváme

$$\Big(\frac{de}{l}\Big)^l \cdot l \cdot e^{-N\left(1-\frac{1}{l}\right)^{l-1}\frac{1}{l}} \leq \Big(\frac{de}{l}\Big)^l \cdot l \cdot e^{-Np^{l-1}(1-p)} \qquad \forall p \in (0,1).$$

Upravujme ještě exponent. Víme, že pro libovolné $l \in \mathbb{N}$ platí

$$\Big(1 - \frac{1}{l}\Big)^l < \frac{1}{e} < \Big(1 - \frac{1}{l}\Big)^{l-1}, \tag{2.19}$$

z čehož ovšem vyplývá

$$-N\Big(1-\frac{1}{l}\Big)^{l-1}\frac{1}{l} < -\frac{N}{e\cdot l}.$$

Nakonec tedy dostáváme

$$P(\exists(j,A)\colon \forall u\in\hat{N}\colon x^u \text{ nesplňuje } (j,A)) \le \Big(\frac{de}{l}\Big)^l \cdot l\cdot e^{-\frac{N}{e\cdot l}}.$$

Položíme-li pak

$$\Big(\frac{de}{l}\Big)^l \cdot l\cdot e^{-\frac{N}{e\cdot l}} < 1,$$

bude z toho plynout

$$0 < P(\forall(j,A)\colon \exists u\in\hat{N}\colon x^u \text{ splňuje } (j,A)),$$

což chceme dokázat. To platí právě tehdy, když

$$el\Big(l\ln\Big(\frac{ed}{l}\Big)+\ln l\Big) < N.$$

Odhad můžeme brát ještě hrubší, když vezmeme N taková, že

$$el^2\Big(\ln d - ln\Big(\frac{l}{e}\Big)\Big) + el\ln l < el^2\Big(\ln d+\frac{\ln l}{l}\Big) \le el^2\Big(\ln d+\frac{1}{e}\Big) = l^2\ln d\Big(e+\frac{1}{\ln d}\Big) \overset{d\ge 2}{\le}$$

$$\overset{d\ge 2}{\le} (2+e)l^2\ln d \le N$$

neboli existuje taková konstanta $C_1 = 2+e$, že pro

$$C_1 l^2\ln d \le N$$

je existence (l,d)-restriction setu již jistá. Jak ale vidíme, jednak jsme udělali mnoho hrubých odhadů, a také přes pravděpodobnostní postup si můžeme být „pouze“ jisti, že taková množina $\{x^1,\dots,x^N\}$ existuje. Avšak existovat může i pro menší N neboli pro infimum přes všechna N taková, že $\{x^1,\dots,x^N\}$ bude (l,d)-restriction set, musí platit

$$R(l,d) \le C_1 l^2\ln d.$$

□

Při bádání o vylepšení odhadu jsme za pomoci pana profesora Vojtěcha Rödla a jeho doktoranda Marcela Tadeu Sales došli ke stejnému výsledku, avšak jinou cestou. Proto si ukažme i tuto alternativu.

Alternativní důkaz. Ukazuje se, že nejlepší vektory tvořící (l,d)-restriciton set jsou takové, ve kterých je přibližně $\frac{d}{l}$ nul (předpokládejme, že $\frac{d}{l}\in\mathbb{N}$) a zbytek jedniček. Mějme tedy permutaci $\sigma\colon\{1,\dots,d\}\to\{1,\dots,d\}$ a uvažme množinu vektorů $\{x^1_\sigma,\dots,x^l_\sigma\}$ takovou, že pro každé $i\in\hat{l}$ a $m\in\hat{d}$ platí:

$$\big(x^i_\sigma\big)_{\sigma(m)} = \begin{cases} 0, & \text{pokud } (i-1)\frac{d}{l}+1 \le m \le i\frac{d}{l}, \\ 1, & \text{pokud } m\notin\{(i-1)\frac{d}{l}+1,\dots,i\frac{d}{l}\}. \end{cases}$$

Pro ilustraci pojmu si uveďme příklad, jak bude vypadat množina vektorů sestrojená v rámci konkrétní identické permutace $\sigma = \mathrm{Id}$:

$$\begin{aligned} x^1_{\mathrm{Id}} &= (\underbrace{0,\dots,0}_{\frac{d}{l}-\text{krát}},1,\dots\dots\dots\dots,1),\\ x^2_{\mathrm{Id}} &= (1,\dots,1,\underbrace{0,\dots,0}_{\frac{d}{l}-\text{krát}},1,\dots\dots\dots\dots 1),\\ &\vdots\\ x^l_{\mathrm{Id}} &= (1,\dots\dots\dots\dots,1,\underbrace{0,\dots,0}_{\frac{d}{l}-\text{krát}}). \end{aligned}$$

Permutace nám pouze pozmění umístění nul a jedniček, avšak jejich počet bude stále stejný. Mějme tedy opět $A \subset \{1,\dots,d\}$ o velikosti $|A| = l-1$ a $j \in \{1,\dots,d\}\setminus A$. Řekneme, že (j,A) je pokryto permutací σ, pokud existuje $u \in \hat{l}$ takové, že $\left(x^u_\sigma\right)\big|_A = 1$ a $\left(x^u_\sigma\right)_j = 0$. Celkem pak máme $\binom{d}{l-1}$ množin A a ke každé z nich $d-l+1$ indexů j, proto existuje celkem $\binom{d}{l-1}(d-l+1)$ možných dvojic (j,A). Naproti tomu pomocí jedné permutace pokryjeme analogicky pouze $\binom{d-\frac{d}{l}}{l-1}\cdot\frac{d}{l}\cdot l$ dvojic (j,A), kde násobíme l kvůli celkově l vektorů sestrojených v rámci jedné permutace. Pokud následně generujeme náhodné $\sigma^1,\dots,\sigma^M$, pak můžeme psát, že

$$p := P((j,A) \text{ je pokryto při } \sigma) = \frac{d\binom{d-\frac{d}{l}}{l-1}}{(d-l+1)\binom{d}{l-1}}.$$

Upravujme však dále

$$p = \frac{(d-l)!(d-\frac{d}{l})!}{(d-1)!(d-\frac{d}{l}-l+1)!} = \frac{(d-\frac{d}{l})(d-\frac{d}{l}-1)\dots(d-\frac{d}{l}-l+3)(d-\frac{d}{l}-l+2)}{(d-1)(d-2)\dots(d-l+2)(d-l+1)}.$$

Jelikož je však funkce

$$f(x) = \frac{d-\frac{d}{l}-x}{d-1-x}$$

klesající pro všechna $x \in [0,d-1)$, tak platí i vztah

$$\frac{d-\frac{d}{l}-j}{d-1-j} \geq \frac{d-\frac{d}{l}-l+2}{d-l+1} \qquad 0 \leq j \leq l-2.$$

Proto můžeme psát

$$p \geq \left(\frac{d-\frac{d}{l}-l+2}{d-l+1}\right)^{l-1}.$$

Snadno pak ověříme, že platí i

$$\frac{d-\frac{d}{l}-l+2}{d-l+1} \geq \frac{l-1}{l},$$

což nám společně s (2.19) dává odhad

$$p \geq \frac{1}{e}.$$

Tudíž při náhodných M permutací můžeme psát, že

$$P(\exists(j,A)\colon (j,A) \text{ není pokryté } \sigma^1,\dots,\sigma^M) \leq \sum_{(j,A)} P((j,A) \text{ není pokryté } \sigma^1,\dots,\sigma^M) =$$

$$= \sum_{(j,A)} \big(1 - P((j,A) \text{ je pokryté } \sigma)\big)^M = \binom{d}{l} l \cdot (1-p)^M \leq \binom{d}{l} l \cdot \left(1 - \frac{1}{e}\right)^M.$$

Jestliže výraz napravo bude ostře menší než jedna, bude to implikovat

$$0 < P(\forall(j,A)\colon (j,A) \text{ je pokryté } \sigma^1,\dots,\sigma^M),$$

čímž bude důkaz hotov. Ovšem z této podmínky plyne, že M musí být takové, aby

$$\ln\left(\binom{d}{l} l\right) + M \ln\left(1 - \frac{1}{e}\right) < 0.$$

Pokud však použijeme odhady (2.9) a (2.17), můžeme upravovat dále, odhad ještě zhoršit a vzít M taková, že

$$\frac{\ln\left(\binom{d}{l} l\right)}{-\ln\left(1-\frac{1}{e}\right)} \overset{(2.9),(2.17)}{\leq} e \ln\left(\frac{(ed)^l}{l^{l-1}}\right) \leq el \ln d + el < M.$$

Z toho plyne, že existuje konstanta $C > 0$ taková, že pro každé $1 \leq l \leq d$ a $d \geq 2$ musí být M alespoň

$$Cl \ln d < M.$$

Ovšem v každé permutaci máme l vektorů, z čehož vyplývá, že při sestrojení $Cl^2 \ln d$ vektorů je existence (l,d)-restriction setu jistá. Proto minimální velikost (l,d)-restriction setu je nanejvýš stejně veliká. Tedy platí, že existuje $C > 0$ taková, že pro každé d a l přirozená, která splňují $1 \leq l \leq d$ a $d \geq 2$, platí:

$$R(l,d) \leq Cl^2 \ln d.$$

□

Věta 2.22. *Existuje taková konstanta $C_2 \in \mathbb{R}^+$, že pro libovolné $l,d \in \mathbb{N}$, které splňují $1 \leq l \leq d$ a $d \geq 2$, platí:*

$$C_2 l \ln d \leq R(l,d).$$

Důkaz. Hlavní myšlenka důkazu je převzata z [23]. Mějme množinu bodů $\{x^1,\dots,x^N\} \subset \{0,1\}^d$ a předpokládejme, že je (l,d)-restriction set. Chceme zkoumat, jaké nejmenší může N být. Vezměme tedy jedlotlivé body $x^1,\dots,x^N$ a udělejme z nich řádky matice X, která je $N \times n$ neboli

$$\forall i \in \hat{N}, \forall j \in \hat{d}\colon \; X_{ij} = 1 \iff (x^i)_j = 1.$$

Když pak vezmeme sloupce matice X a označíme je jako vektory $y^1,\dots,y^d \in \{0,1\}^N$, tak je zřejmé, že

$$\forall i \in \hat{N}, \forall j \in \hat{d}\colon \; (y^j)_i = 1 \iff (x^i)_j = 1.$$

Nakonec ještě definujme pro libovolné $A \subset \{1,\dots,d\}$ taková, že $|A| = l-1$ vektor $y^A = \prod_{i \in A} y^i$, kde násobením vektorů rozumíme po složkách, to znamená

$$\forall j \in \hat{N}\colon \; (y^A)_j = \prod_{i \in A} (y^i)_j = \prod_{i \in A} (x^j)_i. \tag{2.20}$$

Jestliže byl ale $\{x^1, \ldots, x^N\}$ (l,d)-restriction set, tak z definice víme, že pro každé $k \in \{1, \ldots, d\}$ a $A \subset \{1, \ldots, d\} \setminus k$: $|A| = l-1$ existuje $u \in \hat{N}$: $(x^u)_j = 0$ a $(x^u)\big|_A = 1$. Pokud tedy vezmeme libovolné A a B takové, že $A \subset \{1, \ldots, d\}, B \subset \{1, \ldots, d\}, |A| = |B| = l-1$ a $A \neq B$, tak y^A a y^B musí být různé. To plyne z pozorování, že pro libovolné $k \in \{1, \ldots, d\} \setminus A$, a tedy i každé $k \in B \setminus A$, existuje index $j \in \hat{N}$ takový, že $(x^j)\big|_A = 1$ a $(x^j)_k = 0$. Tudíž z (2.20) máme $(y^A)_j = 1$. Jestliže ale pro nějaké $k \in B$ je $(x^j)_k = 0$, pak musí i $(y^B)_j = 0$.

Z toho vyplývá, že pro různé volby A dostáváme různé vektory y^A, z čehož plyne fakt, že zobrazení $A \to y^A$ je prosté. To ovšem implikuje skutečnost, že

$$\#\{A \subset \{1, \ldots, d\}\colon |A| = l-1\} < \#\{y^A\colon y^A \in \{0,1\}^N\}.$$

Vektorů y^A může být celkem 2^N, množin A o velikosti $l-1$ můžeme z d prvků nakombinovat celkem $\binom{d}{l-1}$. Po dosazení můžeme upravit za použití (2.16) na

$$\left(\frac{d}{l-1}\right)^{l-1} \leq \binom{d}{l-1} \leq 2^N,$$

což je ekvivalentní zápisu

$$\frac{1}{\ln 2}(l-1)\ln\left(\frac{d}{l-1}\right) \leq N.$$

Odtud pak můžeme pro velká l a d psát, že existuje $C_2 > 0$ taková, že platí

$$C_2 l \ln d \leq R(l,d)$$

pro každé přirozené l a d splňující $l \leq d$. □

Stejně jako u předešlé věty, i zde jsme došli k alternativnímu důkazu, na který přišel právě Marcelo Tadeu Sales, jemuž tímto patří velké díky. Pro ilustraci jiného přístupu jej zde taktéž uvedeme.

Alternativní důkaz. Mějme množinu $x^1, \ldots, x^N \in \{0,1\}^d$, která je (l,d)-restriction set. Ke každému vektoru x^i, $i \in \hat{d}$ pak uvažme množinu $U^i = \{j \in \{1, \ldots, d\}\colon (x^i)_j = 0\}$, která udává, na kterých složkách je vektor x^i nulový. Sestrojili jsme tedy množiny $U^1, \ldots, U^N$. Když pak zvolíme libovolnou $A \subset \{1, \ldots, d\}$, která je velikosti $|A| = k-1$, tak můžeme ještě zadefinovat množinu indexů

$$I_A = \{i \in \{1, \ldots, N\}\colon U^i \cap A = \emptyset\} \subset \{1, \ldots, N\},$$

která udává indexy vektorů, které mají na jiných složkách, než udává A, nuly. Pokud se však podíváme na sjednocení

$$\bigcup_{i \in I_A} U^i = A^C, \tag{2.21}$$

tak nutně musí dávat komplement množiny A, jelikož přeci pro libovolné A musí pro všechna ostatní j mimo A existovat vektor s nulou na j-té pozici. Z toho však plyne, že pro dané A dostaneme jedinečnou množinu I_A, neboť z rovnice (2.21) jsme sestrojili doplněk množiny A, který je pro různá A jiný. Tudíž zobrazení

$$f\colon A \to I_A$$

je prosté. Množin A je celkem $\binom{d}{l-1}$ a množin I_A opět 2^N, neboť u každého indexu máme dvě možnosti: buď v množině I_A bude, či nebude. Následná argumentace s odvozením je však stejná jako v původním důkazu. □

Pokud tedy zkombinujeme větu 2.19 a 2.21, získáme tím důsledek, jenž je nejdůležitějším poznatkem kapitoly 2.

Důsledek 2.23. *Existuje taková konstanta $C > 0$, že pro libovolné $\varepsilon \in \left(0, \frac{1}{2}\right)$ a $d \in \mathbb{N}, d \geq 2$ platí*

$$C\frac{\ln d}{\varepsilon} \leq N(\varepsilon, d).$$

Důkaz. Vezměme takové $k \in \mathbb{N}$, že nabývá vztah (2.6) a zároveň jsou splněny předpoklady vět 2.19 a předešlé 2.21. Víme, že pak existuje konstanta $C_2 > 0$, pro kterou můžeme psát

$$C_2 2^{k-2} \ln d \leq R\big(2^{k-2}, d\big) \leq N\big(2^{-k}, d\big).$$

Jestliže pak označíme konstantu $\widetilde{C} = \frac{C_2}{4}$, platí

$$\widetilde{C}\frac{\ln d}{\varepsilon} \leq \widetilde{C} 2^k \ln d \leq N\big(2^{-k}, d\big).$$

Nakonec musíme rozmyslet, že funkce $N(\varepsilon, d)$ je pro $\varepsilon \in (0, \frac{1}{2})$ klesající funkce, tudíž

$$N(2^{-k+1}, d) \leq N(\varepsilon, d) \leq N(2^{-k}, d)$$

(pro $\varepsilon \in [\frac{1}{2}, 1]$ je $N(\varepsilon, d) = 1$, neboť stačí brát bod $x = (\frac{1}{2}, \ldots, \frac{1}{2})$). Ovšem pro $N(2^{-k+1}, d)$ bude opět existovat nějaká konstanta $C > 0$, která bude pouhým násobkem konstanty $\widetilde{C}$, že bude platit

$$C\frac{\ln d}{\varepsilon} \leq N(2^{-k+1}, d) \leq N(\varepsilon, d).$$

□

2.3 Srovnání se známými odhady

Pokusme se tedy zmíněné odhady disperze z věty 2.6, respektive důsledku 2.12 a 2.23 porovnat s jinými odhady a zjistit, jestli jsou přelomové nebo už byly překonány. Jak již bylo zmíněno, v článku [22] dokázal Christoph Aistleitner, Aicke Hinrichs a Daniel Rudolf spodní odhad v rovnici (2.2). Z něho pak za využití (2.1) vyplývá, že pro $\varepsilon \in (0, \frac{1}{4})$ a $d \geq 2$, $d \in \mathbb{N}$ platí

$$\frac{1 - 4\varepsilon}{4\varepsilon} \log_2 d \leq N(\varepsilon, d),$$

což je i samotným obsahem [22, Corollary 1]. Jelikož však zároveň pro všechna $\varepsilon \in (0, \frac{1}{4})$ platí

$$\frac{1}{8\varepsilon} < \frac{1 - 4\varepsilon}{4\varepsilon},$$

vyplývá z toho odhad

$$\frac{\log_2 d}{8\varepsilon} \leq N(\varepsilon, d) \quad \forall \varepsilon \in \left(0, \frac{1}{4}\right).$$

Ten je v porovnání s důsledkem 2.23 až na neznámou konstantu zaměnitelný, ovšem my jsme tento odhad rozšířili pro libovolné $\varepsilon \in (0, \frac{1}{2})$, tedy i větší než $\frac{1}{4}$.

Avšak odhad shora přitahuje daleko větší zájem matematiků, proto omezení 2.13 dokázané Mariem Ullrichem a Janem Vybíralem v [21] již bylo vylepšeno. Nejprve Alexander E. Litvak

v [24] přišel s vylepšením, kdy existuje konstanta $C > 0$ taková, že pro libovolné $\varepsilon \in (0, \frac{1}{2})$ a $d \geq 2$, $d \in \mathbb{N}$ platí

$$N(\varepsilon, d) \leq C \log_2(d) \frac{\left(1 + \log_2(\frac{1}{\varepsilon})\right)}{\varepsilon^2}. \tag{2.22}$$

Následně stejný autor ve spolupráci s Galynou V. Livshytsovou v [25] přišli ještě s dalším odhadem, přičemž dokázali pro libovolné $d \in \mathbb{N}$, $d \geq 2$ a $\varepsilon \in (0, \frac{1}{2}]$ odhad

$$N(\varepsilon, d) \leq 12e \frac{4d \ln\ln\left(\frac{8}{\varepsilon}\right) + \ln\left(\frac{1}{\varepsilon}\right)}{\varepsilon}. \tag{2.23}$$

Pokud pak zkombinujeme (2.22) a (2.23), ve stejném článku [25] autoři sestrojili tabulku, která udává zjednodušené výsledky odhadů $N(\varepsilon, d)$ shora v závislosti na různém ε. Zjednodušený zápis pak vypadá následovně:

$$N(\varepsilon, d) \leq \begin{cases} C_1 \ln d \frac{\ln(\frac{1}{\varepsilon})}{\varepsilon^2} & \text{pro} \quad \varepsilon \geq \frac{\ln^2 d}{d \ln\ln(2d)}, \\ C_2 d \frac{\ln\ln(\frac{1}{\varepsilon})}{\varepsilon} & \text{pro} \quad \frac{\ln^2 d}{d \ln\ln(2d)} \geq \varepsilon \geq d^{-d}, \\ C_3 \frac{\ln(\frac{1}{\varepsilon})}{\varepsilon} & \text{pro} \quad d^{-d} \geq \varepsilon \geq d^{-d^2}, \\ C_4 \frac{d^2 \ln d}{\varepsilon} & \text{pro} \quad d^{-d^2} \geq \varepsilon, \end{cases} \tag{2.24}$$

kde C_i pro $i \in \hat{4}$ je vždy jistá kladná konstanta. Posledně zmíněný odhad dokázali Boris Bukh a Ting-Wei Chao v [26].

Poznámka 2.24. V důsledku 2.23 jsme dokázali z věty 2.19 vyvodit a zlepšit spodní odhad $N(\varepsilon, d)$. Zkusme se tedy ještě na problém podívat z druhé strany a pomocí stejné věty a vztahů (2.24) vyvodit naopak odhady na $R(l, d)$. Pokud tedy máme (2.6), respektive $\varepsilon \approx 2^{-k}$ pro jisté $k \in \mathbb{N}$, pak víme z věty 2.19, že existuje konstanta $c > 0$ taková, že

$$R(c2^k, d) \leq N(2^{-k}, d).$$

To má ovšem z definice 2.14 smysl pouze tehdy, když $c2^k < d$. Stejný vztah tedy můžeme vyjádřit pomocí ε jako

$$R\Big(\frac{c}{\varepsilon}, d\Big) \leq N(\varepsilon, d) \qquad \text{pro } \frac{c}{\varepsilon} \leq d. \tag{2.25}$$

Pro $\varepsilon \geq \frac{\ln^2 d}{d \ln\ln(2d)}$ tedy z (2.24) dostáváme

$$R\Big(\frac{c}{\varepsilon}, d\Big) \leq C \frac{\ln d}{\varepsilon^2} \ln\left(\frac{1}{\varepsilon}\right).$$

Ovšem z věty 2.21 víme, že

$$R\Big(\frac{c}{\varepsilon}, d\Big) \leq C \frac{\ln d}{\varepsilon^2},$$

tudíž pro takto velká ε jsme odhad na $R\big(\frac{c}{\varepsilon}, d\big)$ nevylepšili.

Zkusme tedy ještě z (2.24) a (2.25) případ, kdy $\frac{\ln^2 d}{d \ln\ln(2d)} \geq \varepsilon \geq \frac{c}{d}$. Z toho dostáváme odhad

$$R\Big(\frac{c}{\varepsilon}, d\Big) \leq Cd \frac{\ln\ln\left(\frac{1}{\varepsilon}\right)}{\varepsilon},$$

z čehož ovšem ihned plyne, že tento odhad je opět příliš špatný, jelikož

$$1 < \frac{\ln\ln\left(\frac{1}{\varepsilon}\right)}{\varepsilon}$$

a pro $\frac{c}{\varepsilon} \leq d$ platí elementární odhad $R\big(\frac{c}{\varepsilon}, d\big) \leq d$ (viz poznámka 2.15).

Literatura

[1] N. Metropolis, S. Ulam, *The Monte Carlo method.* Journal of the American Statistical Association 44(247), 1949, 335–341.

[2] C. A. Hoar, *Quicksort.* The Computer Journal 5(1), 1962, 10–16.

[3] R. W. Floyd, R. L. Rivest, *Algorithm 489: The algorithm Select - for finding the r-th smallest of n elements.* Communications of the ACM 18(3), 1975, 173.

[4] P. Erdös, *Graph theory and probability.* Canadian Journal of Mathematics 11, 1959, 34–38.

[5] P. Erdös, *Graph theory and probability II.* Canadian Journal of Mathematics 13, 1961, 346–352.

[6] N. Alon, J. H. Spencer, *The Probabilistic Method.* John Wiley & Sons, 2016.

[7] R. Motwani, P. Raghavan, *Randomized Algorithms.* Cambridge University Press, 1995.

[8] G. B. Dantzig, L. R. Ford Jr, D. R. Fulkerson, *A primal-dual algorithm for linear programs.* In H. W. Kuhn, A. W. Tucker (ed.), *Linear Inequalities and Related Systems.* Annals of Mathematics Study 38, 1956, 171–181.

[9] D. R. Karger, *Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm.* Soda 93, 1993, 21–30.

[10] M. Stoer, F. Wagner, *A simple min-cut algorithm.* Journal of the ACM 44(4), 1997, 585–591.

[11] M. I. Shamos, *Computational geometry.* Yale University, 1978.

[12] A. Naamad, D. T. Lee, W.-L. Hsu, *On the maximum empty rectangle problem.* Discrete Applied Mathematics 8(3), 1984, 267–277.

[13] B. Chazelle, R. L. Drysdale, D. T. Lee. *Computing the largest empty rectangle.* SIAM Journal on Computing 15(1), 1986, 300–315.

[14] M. Ullrich, J. Vybíral, *Deterministic constructions of high-dimensional sets with small dispersion.* Algorithmica 84(7), 2022, 1897–1915.

[15] R. Cottle (ed.), *The Basic George B. Dantzig.* Stanford University Press, 2003.

[16] A. Blum, S. Chawla, *Learning from labeled and unlabeled data using graph mincuts.* Carnegie Mellon University, 2001.

[17] F. Estrada, A. Jepson, Ch. Chennubhotla, *Spectral embedding and min cut for image segmentation.* In British Machine Vision Conference, London, 2004, 317–326.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms.* MIT Press, 2001.

[19] T. Kesselheim, K. Mehlhorn, *Min cut, fast cut, polynomial identities.* [online]. [cit. 20.3.2023]. Dostupné z: `https://www.mpi-inf.mpg.de/fileadmin/inf/d1/teaching/summer16/random/MinCutFastCutPolynomialIdentities.pdf`.

[20] J. Edmonds, J. Gryz, D. Liang, R. J. Miller, *Mining for empty spaces in large data sets.* Theoretical Computer Science 296(3), 2003, 435–452.

[21] M. Ullrich, J. Vybíral, *An upper bound on the minimal dispersion.* Journal of Complexity 45, 2018, 120–126.

[22] C. Aistleitner, A. Hinrichs, D. Rudolf, *On the size of the largest empty box amidst a point set.* Discrete Applied Mathematics 230, 2017, 146–150.

[23] D. J. Kleitman, J. Spencer, *Families of k-independent sets.* Discrete mathematics 6(3), 1973, 255–262.

[24] A. E. Litvak, *A remark on the minimal dispersion.* Communications in Contemporary Mathematics 23(06), 2021, 2050060.

[25] A. E. Litvak, G. V. Livshyts, *New bounds on the minimal dispersion.* Journal of Complexity 72, 2022, 101648.

[26] B. Bukh, T.-W. Chao, *Empty axis-parallel boxes.* International Mathematics Research Notices 2022(18), 2022, 13811–13828.