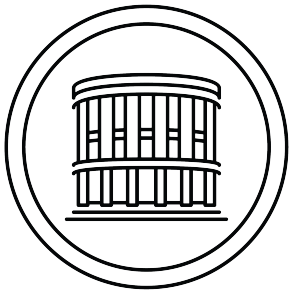
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



GENEROVANIE NÁHODNÝCH ČÍSEL POMOCOU VEĽKÝCH JAZYKOVÝCH MODELOV UMELEJ INTELIGENCIE

DIPLOMOVÁ PRÁCA

2025 Vendelín Michal HOZZÁNK

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

GENEROVANIE NÁHODNÝCH ČÍSEL POMOCOU VEĽKÝCH JAZYKOVÝCH MODELOV UMELEJ INTELIGENCIE

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomicko-finančná matematika a modelovanie
Študijný odbor:	Matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	doc. Mgr. Richard Kollár, PhD.

Bratislava 2025 **Vendelín Michal HOZZÁNK**





Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Vendelín Michal Hozzánk
Študijný program: ekonomicko-finančná matematika a modelovanie (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: matematika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Generovanie náhodných čísel pomocou veľkých jazykových modelov umelej inteligencie
Generating random numbers using the large language models of artificial intelligence

Anotácia: Veľké jazykové modely (LLM) umelej inteligencie (UI) v mnohých otázkach dokážu dávať odpovede porovnateľné s ľuďmi, či s výkonnými deterministickými alebo stoachastickými algoritmami, prípadne ich aj prevyšovať. Zároveň však preukazujú veľkú mieru generovania odpovedí, ktoré sa len javia správnymi či relevantnými, a obsahujú veľké množstvo biasovaných alebo náhodne generovaných informácií. Práca sa bude zaoberať štúdiom toho, ako v rôznych kontextoch dokážu tieto nástroje UI generovať náhodné čísla.

Vedúci: doc. Mgr. Richard Kollár, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: doc. Mgr. Igor Melicherčík, PhD.

Dátum zadania: 12.01.2024

Dátum schválenia: 25.01.2024 prof. RNDr. Daniel Ševčovič, DrSc.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné vyhlásenie Čestne vyhlasujem, že celú diplomovú prácu na tému "Generovanie náhodných čísel pomocou veľkých jazykových modelov umelej inteligencie", vrátane všetkých jej príloh a obrázkov, som vypracoval samostatne, a to s použitím literatúry uvedenej v priloženom zozname a nástrojov umelej inteligencie. Vyhlasujem, že nástroje umelej inteligencie som použil v súlade s príslušnými právnymi predpismi, akademickými právami a slobodami, etickými a morálnymi zásadami za súčasného dodržania akademickej integrity a že ich použitie je v práci vhodným spôsobom označené.

Miesto a dátum: ..... Titul, meno a priezvisko: .....

Poďakovanie Touto cestou sa chcem poďakovať môjmu vedúcemu diplomovej práce doc. Mgr. Richardovi Kollárovi, PhD. za obrovskú trpezlivosť, ochotu a pomoc počas celého procesu písania, veľmi podnetné pripomienky, ktoré mi pomohli pri písaní tejto práce. Ďakujem aj svojej partnerke a rodine za ich podporu a trpezlivosť.

Abstrakt v štátnom jazyku

HOZZÁNK, Vendelín Michal: *Generovanie náhodných čísel pomocou veľkých jazykových modelov umelej inteligencie*, Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: doc. Mgr. Richard Kollár, PhD., Bratislava, 2025, 89 s.

Táto práca sa zaoberá skúmaním techník interakcie s veľkými jazykovými modelmi (LLM) umelej inteligencie, pričom ako experimentálny nástroj využíva generovanie náhodných čísel. Cieľom práce je identifikovať správne prístupy k efektívnej komunikácii s LLM a odhaliť ich limity pri riešení úloh, na ktoré neboli primárne navrhnuté. Náhodné čísla boli zvolené ako ideálny prípadový nástroj, pretože vyžadujú precízne usmernenie modelov, pričom ich kvalitu je možné presne merať pomocou explicitných a implicitných testov.

V rámci prompt engineeringu sme definovali tri úrovne komunikácie (`L1`, `L2`, `L3`), v ktorých sme postupne pridávali nové techniky. Pre vyhodnotenie kvality generovaných sekvencií sme implementovali ucelený testovací rámec: explicitné štatistické testy a implicitné scenáre z oblasti optimalizácie. Výsledky ukázali, že žiaden z datasetov generovaných GPT4 nepreukazuje náhodné správanie, no stratégia `L3` významne znižuje systematické biasy. Implicitné testovanie potvrdilo, že vhodné promptovanie má zásadný vplyv na rýchlosť a konzistentnosť konvergencie optimalizačných algoritmov. Navyše sme náhodou zistili, že dataset `L1`, hoci obsahuje opakujúce sa vzory, prináša v počiatočných iteráciách Simulovaného žíhania nečakané zrýchlenie procesu optimalizácie.

Kľúčové slová: Veľké jazykové modely, Náhodnosť, Prompt engineering, Štatistické testovanie, Umelá inteligencia

Abstract

HOZZÁNK, Vendelín Michal: *Generating random numbers using the large language models of artificial intelligence*, Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: doc. Mgr. Richard Kollár, PhD., Bratislava, 2023, 89 p.

This thesis investigates techniques for interacting with large language models (LLMs) of artificial intelligence, using random number generation as an experimental testbed. Its aim is to identify effective approaches to communicating with LLMs and to reveal their limitations on tasks for which they were not originally designed. Random numbers were chosen as the ideal case study because they require precise model guidance and their quality can be rigorously measured through both explicit statistical tests and implicit optimization scenarios.

Within the scope of prompt engineering, we defined three communication levels (`L1`, `L2`, `L3`), incrementally incorporating new techniques at each stage. To evaluate the quality of the generated sequences, we implemented a comprehensive testing framework: a suite of explicit statistical tests alongside implicit scenarios drawn from optimization problems. Our results show that none of the datasets generated by GPT4 exhibit truly random behavior, although the `L3` strategy substantially reduces systematic biases. Implicit testing further confirms that appropriate prompt engineering has a decisive impact on the speed and consistency of convergence in optimization algorithms. Incidentally, we also discovered that the `L1` dataset—despite its repeating patterns—yields an unexpected acceleration in the early iterations of simulated annealing.

Keywords: Large language models, Randomness, Prompt engineering, Statistical testing, Artificial intelligence

Obsah

Úvod **8**

1 Úvod do veľkých jazykových modelov a náhodnosti **10**

1.1 História a vývoj 10

1.2 Náhodnosť 12

2 Návrh a realizácia experimentu **15**

2.1 Generovanie datasetov 15

2.1.1 Počiatočné naivné pokusy a ich neúspechy 16

2.1.2 Poznámka k použitému jazyku v konverzáciách s GPT4 17

2.1.3 Použité techniky promptovania 18

2.1.4 Generovanie dátových rámcov 21

2.2 Explicitné testy 26

2.3 Implicitné experimenty 32

2.3.1 Simulované žíhanie (SA) ako metóda voľnej optimalizácie spojitej funkcie 33

2.3.2 Úloha Knapsack (batoh) s využitím SA 36

3 Testovanie čísel generovaných LLM **39**

3.1 Explicitné testovanie generovaných postupností 39

3.1.1 Porovnanie `L2` a `L2mod` – Anomália 39

3.1.2 Porovnanie `L2` a `L2mod` – Vizualizácie a výsledky testov 40

3.1.3 Porovnanie všetkých finálnych datasetov 54

3.2 Implicitné testovanie generovaných postupností 72

3.2.1 Simulované žíhanie (SA) 72

3.2.2 Knapsack 79

3.3 Diskusia k výsledkom 81

Záver **84**

Zoznam použitej literatúry **86**

Úvod

Nástroje umelej inteligencie, konkrétne veľké jazykové modely (LLM), v posledných rokoch zásadne zmenili spôsob, akým pristupujeme k analýze textu, modelovaniu prirodzeného jazyka a riešeniu komplexných problémov. Tieto modely, ako napríklad GPT4, ukázali pozoruhodnú schopnosť generovať súvislé texty, odpovedať na zložité otázky a dokonca riešiť úlohy, na ktoré neboli pôvodne určené. Napriek týmto pokrokom zostáva otázkou, ako efektívne a presne komunikovať s LLM tak, aby človek dosiahol požadovaný výsledok. Táto schopnosť "rozprávať sa" s modelmi – teda vytváranie správnych príkazov a otázok, známe aj ako prompt engineering – je kľúčovým faktorom pre ich úspešné využitie v širokom spektre aplikácií.

Cieľom tejto práce nie je samotné generovanie náhodných čísel pomocou LLM, ale skôr skúmanie techník a postupov, ktoré umožňujú efektívnu interakciu s modelmi. Náhodné čísla slúžia ako ideálny testovací nástroj, pretože sú výzvou, ktorá presahuje prirodzené schopnosti týchto modelov. LLM neboli vytvorené na generovanie náhodnosti a často sa pri tejto úlohe správajú nepredvídateľne alebo odmietajú splniť požiadavky priamo. To vytvára potrebu využívať sofistikované prístupy k ich "presviedčaniu", ktoré sa následne dajú aplikovať v mnohých iných oblastiach využitia umelej inteligencie.

Aby sme mohli získať maximálnu hodnotu z veľkých jazykových modelov, je potrebné pochopiť nielen ich schopnosti, ale aj obmedzenia. Ich obmedzenia je možné rozpoznať aj z ich histórie a vývoja v Kapitole 1, alebo rozsiahleho tela výskumu o ich vlastnostiach. Prehĺbenie poznania o tom, ako sa LLM vyvinuli od jednoduchých štatistických jazykových modelov po dnešné sofistikované architektúry, ponúka potrebné teoretické zázemie pre ich využitie na špecifické účely, ako je napríklad generovanie náhodných čísel.

Ďalším dôvodom, prečo je generovanie náhodných čísel vhodnou voľbou pre skúmanie správania LLM, je možnosť presného merania ich výstupov. Z dôsledného definovania náhodnosti v Podkapitole 1.2 prirodzene vyplývajú testy na jej prítomnosť a kvalitu. Táto vlastnosť dát sa dá hodnotiť pomocou explicitných či implicitných testov, ako možno vidieť v Podkapitolách 2.2 a 2.3. Podkapitola 2.2 obsahuje nielen adaptácie už existujúcich štandardných testov, ale aj nami dizajnované testy. Hodnotenie náhodnosti pomocou už spomínaných testov umožňuje objektívne posúdiť, do akej miery sa výstupy modelov približujú požadovaným vlastnostiam.

Po vytvorení pevného teoretického základu sa práca v Podkapitole 2.1 zameriava na proces, ako prinútiť LLM generovať náhodné čísla. Tento proces zahŕňa nielen rôzne techniky prompt engineeringu, ale aj praktické skúsenosti z iteratívneho skúšania a robenia chýb. Dokumentuje sa, čo fungovalo, čo nie, a aké úpravy boli potrebné, aby sa dosiahol žiadaný výsledok. Táto analýza odhaľuje, ako modely reagujú na rôzne typy príkazov, a zdôrazňuje potrebu adaptácie prístupu na základe spätnej väzby z modelu.

Nakoniec, v Kapitole 3, sa práca venuje samotnej aplikácii vytvoreného testovacieho balíka na generované čísla. Táto fáza zahŕňa analýzu výsledkov a porovnanie kvality výstupov LLM s tradičnými pseudonáhodnými generátormi. Výsledky poskytujú nielen vhľad do toho, ako dobre modely plnia túto úlohu, ale aj cenné poznatky o tom, ako pristupovať k ďalším úlohám, ktoré vyžadujú jemné ladenie interakcie s LLM.

Táto práca teda nie je len experimentom s generovaním náhodných čísel, ale aj prieskumom metodológií, ktoré môžu byť široko aplikované na rôzne aspekty práce s umelou inteligenciou. Jej cieľom je ponúknuť rady na efektívnu interakciu s LLM, podložené systematickým výskumom a praktickými skúsenosťami, ako aj odhaliť a pomenovať nedostatky súčasných verzií LLM pri riešení úloh, na ktoré neboli primárne navrhnuté.

1 Úvod do veľkých jazykových modelov a náhodnosti

V tejto kapitole najprv predstavíme kontext a postupný vývoj jazykových modelov od tradičných štatistických prístupov v Podkapitole 1.1 cez prelomové neurónové architektúry až po dnešné transformerom riadené veľké jazykové modely so státisícami miliárd parametrov. Následne sa zameriavame na problematiku náhodnosti v Podkapitole 1.2: definujeme rozdiel medzi skutočnou náhodnosťou a pseudonáhodnosťou, stručne rozoberieme ľudskú pochopenie náhodnosti aj naše očakávania od LLM pri generovaní náhodných sekvencií.

1.1 História a vývoj

Vývoj veľkých jazykových modelov (LLM) je úzko spätý s históriou spracovania jazyka (NLP). Tento proces prešiel niekoľkými zásadnými zmenami, od štatistických prístupov cez neurónové modely až po dnešné modely s obrovskou kapacitou a schopnosťou vykonávať niektoré úlohy na úrovni blízko ľudskej. Táto kapitola skúma vývoj jazykových modelov cez jeho významné fázy: štatistické jazykové modely (SLM) a neurónové jazykové modely (NLM), ktoré vyvrcholili v LLM. Každá z týchto etáp priniesla nové technológie a prístupy, ktoré prekonali limity svojich predchodcov.

1.1.1 Štatistické jazykové modely (SLM)

SLM [1] sú jedným z prvých nástrojov používaných v NLP na analýzu a generovanie prirodzeného jazyka. Ich základom je štatistická analýza textových dát, kde každé slovo alebo sekvencia slov je reprezentovaná ako náhodná premenná. Pravdepodobnosti výskytu slov sa určujú na základe frekvencie ich výskytu v trénovacej sade. Typickým príkladom sú n-gramové modely, kde pravdepodobnosť každého slova závisí len od obmedzeného počtu predchádzajúcich slov, ako napríklad v bigramoch (jedno predchádzajúce slovo) alebo trigramoch (dve predchádzajúce slová).

SLM boli užitočné pre úlohy, ako je predikcia textu a jednoduché jazykové úlohy, no mali svoje limity. Hlavnou nevýhodou bola ich neschopnosť modelovať dlhodobejšie závislosti medzi slovami, keďže pravdepodobnosť slov sa počítala len na základe veľmi obmedzeného kontextu. Tento problém viedol k častej strate významových vzťahov medzi

vzdialenejšími slovami v texte. Ďalším problémom boli nulové pravdepodobnosti pre sekvencie, ktoré sa v trénovacích dátach nevyskytli, čo vyžadovalo zavedenie zhladzovacích techník na zmiernenie tohto efektu.

SLM síce ponúkli základ pre NLP, ale neboli schopné efektívne zachytiť zložité štruktúry a nuansy prirodzeného jazyka. Ich obmedzenia, vrátane štatistického prístupu založeného na frekvenciách, ukázali potrebu sofistikovanejších metód, čo viedlo k vývoju NLM.

1.1.2 Neurónové jazykové modely (NLM)

NLM [1] predstavujú významný krok vpred v oblasti modelovania prirodzeného jazyka. Ich základným princípom je využitie neurónových sietí na zachytenie komplexných vzorov a vzťahov v jazyku. Takisto dokážu efektívne spracovať dlhšie kontexty a lepšie pochopiť sémantické a syntaktické nuansy textu.

Jedným z kľúčových konceptov v NLM sú vektory slov (word embeddings). Tieto vektory reprezentujú jednotlivé slová ako numerické hodnoty v mnohorozmernom priestore, pričom slová s podobným významom sú v tomto priestore bližšie k sebe. Napríklad vektor pre slovo "mačka" bude bližší vektoru pre "pes" ako vektoru pre "strom". Tento prístup umožňuje modelu pochopiť a využiť významové vzťahy medzi slovami. Najznámejšie techniky na vytváranie vektorov slov, ako je Word2Vec alebo GloVe, sú základom mnohých moderných NLM.

Jednou z hlavných výhod NLM je ich schopnosť generalizovať. Na rozdiel od SLM, ktoré sa spoliehajú na frekvencie n-gramov, NLM dokážu zachytiť vzťahy medzi slovami aj v prípadoch, keď sa konkrétna kombinácia slov v trénovacích dátach nevyskytla. Napríklad ak sa model naučil vzťah medzi "mačka" a "loví" a zároveň medzi "pes" a "loví", dokáže zovšeobecniť, že aj "mačka" a "pes" majú podobné kontexty (vzdialenosti v priestore). Táto schopnosť je kľúčová pri spracovaní jazykov s bohatou morfológiou alebo pri práci s limitovaným množstvom trénovacích dát.

Napriek svojim výhodám NLM čelia aj výzvam. Ich tréning vyžaduje veľké množstvo dát a výpočtových zdrojov, čo môže byť nákladné. Pri NLM nastáva aj problém s interpretovateľnosťou, keďže ich komplexné architektúry sťažujú pochopenie toho, ako model dospel k svojim predikciám. Tento problém sa iba prehlbuje v neskorších modeloch...

V konečnom dôsledku NLM predstavujú revolučný posun v spracovaní prirodzeného

jazyka v porovnaní s SLM. Ich schopnosť zachytiť hlbšie vzory a vzťahy v jazyku položila základy pre ešte pokročilejšie modely, ako sú predtrénované jazykové modely (PLM) a veľké jazykové modely (LLM). Tieto ďalšie generácie modelov priniesli ešte vyššiu presnosť a adaptabilitu do širokého spektra aplikácií.

1.1.3 Veľké jazykové modely (LLM)

Veľké jazykové modely predstavujú súčasný vrchol v evolúcii metód spracovania prirodzeného jazyka. Tieto modely sú založené na transformerovej architektúre a využívajú overenú stratégiu dvojfázového učenia – rozsiahly predtréning na obrovských korpusoch textu, po ktorom nasleduje prípadné doladenie na konkrétne úlohy. Čo však LLM skutočne odlišuje od predchádzajúcich generácií, je ich masívna kapacita: bežne obsahujú desiatky miliárd parametrov a trénujú sa na stovkách gb až tb dát. Vďaka tomu dokážu zachytiť veľmi jemné a dlhodobé závislosti v texte, reagovať na skutočne rôznorodé požiadavky a na mnohé úlohy ponúknuť riešenie ihneď po vhodne navrhnutom prompte, bez potreby ďalšieho finetuningu.

Napriek svojej pozoruhodnej univerzálnosti sa však LLM stretávajú aj s významnými výzvami. Tréning a nasadenie týchto modelov si vyžaduje značné výpočtové zdroje a často špecializovaný hardvér, čo obmedzuje ich dostupnosť v menších organizáciách. Navyše, modely tohto rozsahu občas generujú takzvané „halucinácie“ – pútavé, no nepravdivé či nekonzistentné tvrdenia, ktoré môžu byť nebezpečné v kontextoch vyžadujúcich vysokú spoľahlivosť. Kľúčovým problémom zostáva aj interpretovateľnosť: čím väčší model, tým ťažšie dokážeme sledovať, ktoré parametre a vstupy viedli k danému výstupu. Napriek týmto limitáciám však LLM jednoznačne zmenili oblasť NLP, pretože dokážu riešiť širokú škálu jazykových úloh s kvalitou blížiacou sa ľudskej úrovni.

1.2 Náhodnosť

Predstavíme skutočnú náhodnosť (napríklad fyzikálne procesy ako rádioaktívny rozpad alebo atmosférický šum) oproti pseudo-náhodnosti (generovanej deterministickými algoritmami, ako je Mersenne Twister a iné) a následne aj ľudské porozumenie náhodnosti. Nakoniec spíšeme naše očakávania náhodnosti pre LLM.

Skutočná náhodnosť

Skutočná náhodnosť je charakteristická javmi, ktoré sú výsledkom nepredvídateľných fyzikálnych procesov [2]. Tieto procesy sú nedeterministické, čo znamená, že ich výstupy nemožno presne predpovedať ani pri úplnej znalosti vstupných podmienok. Medzi najčastejšie zdroje skutočnej náhodnosti patria rádioaktívny rozpad, pri ktorom je čas rozpadu jednotlivých atómov úplne náhodný, alebo atmosférický šum. Tieto procesy poskytujú vysoko kvalitné náhodné čísla, ktoré sú využiteľné v oblastiach, kde je dôležitá absolútna nepredvídateľnosť, ako je kryptografia alebo bezpečnostné aplikácie. Ich generovanie je však technicky a časovo náročné, čo limituje ich dostupnosť v iných praktických aplikáciách.

Pseudonáhodnosť

Pseudonáhodnosť sa na rozdiel od skutočnej náhodnosti zakladá na deterministických algoritmoch, ktoré simulujú náhodné správanie. Generátory pseudonáhodných čísel (PRNG) vytvárajú sekvencie hodnôt, ktoré na prvý pohľad pôsobia náhodne, no v skutočnosti sú úplne deterministické, ak poznáme počiatočné podmienky (tzv. seed) [2]. Jedným z najrozšírenejších PRNG je Mersenne Twister [3], ktorý vyniká dlhou periódou a veľmi dobrou kvalitou generovaných sekvencií, avšak ani on sa nevyhne určitým štruktúram pri dostatočne veľkom počte vygenerovaných čísel.

Hlavným obmedzením pseudo-náhodných čísel je ich deterministická povaha. V momente, keď je seed zistený alebo odhadnutý, dokážeme reprodukovať všetky následné hodnoty, čo môže byť kritické najmä v kontexte kryptografických aplikácií. V praxi preto pseudo-náhodné čísla nachádzajú široké uplatnenie v simuláciách, hrách alebo štatistických analýzach, kde úplná náhodnosť nie je potrebná.

Naopak, v kritických aplikáciách (kryptografia alebo citlivé simulácie) uprednostňujeme hardvérové alebo kryptograficky bezpečné generátory, ktoré zaručujú vyššiu úroveň nepreniknuteľnosti a kvality entropie.

Ľudské pochopenie náhodnosti

Ľudské chápanie náhodnosti je často intuitívne a nezodpovedá matematickým alebo fyzikálnym definíciám [4]. Ľudia majú tendenciu vykazovať pravidelnosť pri manuálnom generovaní náhodných sekvencií, čo vedie k častým chybám, ako je preferencia určitých vzorov alebo vyvarovanie sa opakovaniu podobných hodnôt. Tento jav sa prejavuje napríklad pri hádaní výsledkov hodu mincou, kde ľudia majú tendenciu vyhýbať sa opakovaniu rovnakého výsledku, hoci je to z pohľadu skutočnej náhodnosti úplne legitímne. Tieto kog-

nitívne skreslenia zdôrazňujú rozdiel medzi ľudským intuitívnym chápaním náhodnosti a jej matematickou podstatou, čo má praktické dôsledky napríklad v oblastiach, kde sa ľudia spoliehajú na svoje subjektívne odhady náhodných javov.

Očakávané správanie LLM pri generovaní náhodnosti

Keď majú LLM za úlohu generovať náhodné čísla bez prístupu k externým knižniciam alebo programovacím jazykom, môžeme očakávať, že budú pravdepodobne premietať do svojich výstupov mechanizmy naučené z rozsiahlych textových korpusov. Očakávame, že budú mať rovnakú tendenciu k vyhýbaniu sa "nepríjemným" vzorom, akú majú ľudia pri manuálnom písaní náhodných čísel. Ako je už písané skôr, ľudia prirodzene minimalizujú dlhé sekvencie rovnakých číslic, snažia sa meniť cifry a často sa zdráhajú použiť tri či viac rovnakých symbolov za sebou.

Práve tieto "ľudské" návyky by mohli preniknúť aj do výstupov LLM. Model totiž neoperuje so skutočnou numerickou náhodou, ale predpovedá najpravdepodobnejší ďalší token na základe vzorov vo svojom tréningovom texte. Ak sa v textoch vyskytujú reťazce ako "123123" alebo ak autori textov pri opisovaní náhodných príkladov preferujú určité cifry či vyhýbajú sa príliš dlhým opakovaniam, model tieto sklony pravdepodobne zopakuje.

V tejto práci sa preto zameriame nielen na meranie celkovej uniformity a nezávislosti cifier, ale aj na detekciu takýchto javov. Budeme skúmať, do akej miery sa LLM vyhýbajú opakovaniam, či vytvárajú vzory pripomínajúce ľudské ťažkosti pri simulovaní náhodnosti, a ako sa tieto javy líšia od správania klasických pseudonáhodných generátorov. Takéto porovnanie nám umožní pochopiť, kde modely čerpajú svoje predstavy o náhodnosti – či skutočne z "matematického" rozdelenia, alebo či zostávajú verné svojmu jazykovo orientovanému pôvodu.

2 Návrh a realizácia experimentu

LLMs neboli primárne navrhnuté na generovanie náhodných čísel. Ako sme už spomínali v predchádzajúcej kapitole, ich fungovanie je založené na pravdepodobnostnom predpovedaní nasledujúceho tokenu na základe kontextu, čo znamená, že skutočná náhodnosť v tradičnom štatistickom zmysle nie je ich prirodzenou vlastnosťou. Naopak, LLM majú tendenciu vyhľadávať vzory, opakovať určité sekvencie a prispôsobovať svoje výstupy očakávaniam vyplývajúcim z ich trénovacích dát. Z tohto dôvodu je potrebné pristupovať k ich využitiu na generovanie náhodných čísel s rozvahou a vhodne zvolenými stratégiami.

Prvým cieľom tejto kapitoly je preskúmať, akými spôsobmi je možné modely "presvedčiť", aby generovali výstupy, ktoré sa čo najviac približujú náhodným hodnotám, a aké výzvy sú s tým spojené. Tomuto cieľu sa práca venuje v Podkapitole 2.1. Druhým cieľom je predstaviť mechanizmy testov a experimentov (explicitných a implicitných), ktoré budú náhodnosť testovať a skúmať. Testom a experimentom sa venujú Podkapitoly 2.2 a 2.3.

2.1 Generovanie datasetov

Jedným z hlavných problémov pri práci s LLMs je ich inherentná "neochota"generovať skutočne náhodné čísla, čo si vyžaduje experimentovanie s rôznymi technikami promptovania. Táto kapitola sa venuje analýze efektivity rôznych prístupov – od jednoduchých požiadaviek na generovanie čísiel až po komplexnejšie stratégie obchádzania predpísaných limitácií cloud-based modelov. Skúmame, ako zmeniť formuláciu promptov tak, aby model generoval čísla s dostatočným počtom desatinných miest, ako ho presvedčiť na generovanie väčších dátových súborov naraz a aké postupy sú potrebné na správne nadväzovanie promptov. Na základe praktických experimentov popisujeme, ktoré prístupy boli úspešné a ktoré sa ukázali ako neefektívne, čím poskytujeme ucelený pohľad na techniky získavania dát z LLM v kontexte náhodnosti.

Pre naše experimenty sme zvolili model GPT4, pretože v súčasnosti predstavuje jednu z najpoužívanejších a najdôveryhodnejších platforiem pre generovanie textu na svete. Vďaka svojej vysokej presnosti a hlbokému porozumeniu kontextu dokáže GPT4 často poskytovať konzistentné a koherentné reakcie aj na neštandardné požiadavky. Okrem

toho disponuje vysokým limitom tokenov, čo nám umožnilo generovať rozsiahlejšie datasety naraz, a je predmetom rozsiahleho akademického výskumu aj výskumu v praxi, čím zabezpečuje dostatok porovnávacích štúdií a overených poznatkov.

Po tejto podkapitole budeme mať k dispozícii päť datasetov, z toho tri generované pomocou LLM a dva pomocou PRNG. Podkapitola začína však tam, kde sme začínali aj my.

2.1.1 Počiatočné naivné pokusy a ich neúspechy

Pri našich prvých pokusoch mal GPT4 tendenciu ihneď pristúpiť k použitiu Pythonu. Tejto tendencii sme sa snažili vyhnúť. Pre potreby testov náhodnosti a experimentov na našich datasetoch popísaných v Podkapitolách 2.2 a 2.3 sme sa snažili donútiť GPT4 generovať čísla z intervalu $[0; 1]$. Ako stratégiu na donútenie GPT4 generovať čísla vlastným algoritmom a nie pomocou PRNG sme mu zakaždým nariadili generovať čísla s 16 desatinnými miestami. Táto stratégia nám zvýšila úspešnosť vyhnutiu sa PRNG. Na Obrázku 1 možno vidieť začiatok prvej (neúspešnej) konverzácie s GPT4. Konverzáciu nazývame neúspešnou z dôvodu použitia Pythonu.

> *I need a table of random numbers between 0 and 1, each number with 16 decimal places. I wish for this table to have 1000 such values. Can you export it as a csv file please? I'd like to download it afterwards.*

> **GPT-4:** [Python code] The file with 1000 random numbers, each with 16 decimal places, has been generated. You can download it using the link below: [link].

Obr. 1: Ukážka prvého naivného promptu používateľa a odpovede GPT4.

Náš ďalší pokus pozostával v dramatickom zvýšení počtu požadovaných náhodných čísel s presným formátom. Prvýkrát sme tu videli chybu, respektíve nepravdu, ktorú napísal. Síce nám napísal, že generované čísla majú 16 desatinných pozícií, no v skutočnosti ich mali iba 15. Môžeme ho vidieť na Obrázku 2.

*I need a table of random numbers between 0 and 1, each number with 16 decimal places. I wish for this table to have **20 000** such values. Can you export it as a CSV file please? I'd like to download it afterwards.*

GPT4: [Python code] I've created a CSV file with 20 000 random numbers between 0 and 1, each with 16 decimal places. You can download it using the link below: [link].

Obr. 2: Ukážka promptu používateľa so žiadosťou o 20 000 čísel a odpovede GPT4.

Po upozornení na tento fakt GPT4 túto nezrovnalosť opravil a vygeneroval 20 000 čísel s 16 desatinnými miestami. Po aplikácii podskupiny textov predstavených v Podkapitole 2.2 nám vyšlo, že 16. pozícia má dramaticky iné rozdelenie než predchádzajúce. Prvých 15 pozícií malo správanie konzistentné s použitím PRNG, no 16. nie. Po bližšom preskúmaní datasetu vyšlo najavo, že GPT4 prepoužil už vygenerované čísla z predošlého promptu a iba na ich koniec pridal jednu cifru. Podľa našich testov táto pridaná cifra nebola generovaná pomocou PRNG, čiže ju asi vygeneroval pomocou vlastného jazykového modelu.

Ďalej predstavíme techniku generovania výsledných datasetov, ku ktorej sme sa dostali po prečítaní literatúry k tejto problematike a viacerých neúspešných experimentoch s GPT4.

2.1.2 Poznámka k použitému jazyku v konverzáciách s GPT4

Ako môžeme vidieť z predchádzajúcich príkladov konverzácie na Obrázkoch 1 a 2, vo všetkých interakciách s veľkými jazykovými modelmi (v našom prípade GPT4) sme používali anglický jazyk. Dôvodom nie je len pohodlie alebo konzistencia v komunikácii, ale najmä výrazne lepšia empiricky preukázaná kvalita odpovedí modelov práve v angličtine. Táto skutočnosť vyplýva predovšetkým zo spôsobu, akým sú tieto modely trénované.

Podľa správ od vývojárov ako OpenAI [7], je drvivá väčšina tréningových dát pre súčasné LLM (ako GPT3, GPT4) práve v anglickom jazyku. To znamená, že anglický

jazyk je pre tieto modely dominantný nielen kvantitou, ale aj kvalitou a rôznorodosťou dostupných textov, čo umožňuje lepšie pochopenie jemných nuáns a vyššiu spoľahlivosť výstupov. Naopak, pri menej zastúpených jazykoch, ako je aj slovenčina, môže byť presnosť generovaných odpovedí znížená, pretože modely disponujú výrazne menším tréningovým súborom, čo obmedzuje ich schopnosť správne pochopiť jemné nuansy či špecifické kontexty.

Pri komunikácii v anglickom jazyku LLM dosahuje vyššiu kvalitu výstupov, lepšiu konzistentnosť a presnosť odpovedí, ako aj vyššiu schopnosť správne interpretovať zadaný prompt. Pri práci so slovenskými alebo inými menej rozšírenými jazykmi sa často prejavujú chyby v interpretácii príkazov, horšia konzistentnosť a menej presné odpovede, čo môže byť výraznou nevýhodou, predovšetkým pri úlohách, ktoré vyžadujú vysokú presnosť a reprodukovateľnosť, ako je práve generovanie náhodných čísel.

Z tohto dôvodu je pri výskume a pri aplikáciách, kde záleží na presnosti výsledkov, odporúčané komunikovať s LLM primárne v anglickom jazyku. To platí najmä v prípadoch, ak potrebujeme dosiahnuť vysokú úroveň kontroly nad výstupom modelu. Táto práca preto volí angličtinu ako jazyk promptovania.

Napriek tomu je potrebné zdôrazniť, že komunikácia v slovenskom jazyku môže byť vhodná v prípadoch, kde je prioritou prirodzenosť a pohodlnosť interakcie, a kde nižšia presnosť výstupov nespôsobuje zásadné problémy. Všetky nasledujúce praktické experimenty a interakcie s GPT4 budú preto realizované v anglickom jazyku, čím sa zabezpečí optimálna presnosť, spoľahlivosť a objektivita získaných výsledkov.

2.1.3 Použité techniky promptovania

Pri posudzovaní literatúry použitej v tejto časti práce je dôležité zdôrazniť, že niektoré citované štúdie sú dostupné iba ako preprinty, čo odráža dynamický a rýchlo sa meniaci charakter výskumu v oblasti veľkých jazykových modelov a promptového inžinierstva. Tento prístup umožňuje autorom zahrnúť najnovšie poznatky a trendy, aj keď tieto štúdie ešte neprešli celým recenzným procesom. Takáto flexibilita je nevyhnutná pre adekvátne zachytenie rýchleho tempa vývoja v tejto oblasti, pričom prispieva k neustálemu zlepšovaniu a optimalizácii modelov, ako je práve napríklad GPT4.

(1) Presnosť: Výskum ukazuje, že jasne formulované a podrobné prompty môžu

výrazne zlepšiť výkon jazykových modelov. Presná terminológia a explicitné pokyny umožňujú modelom lepšie pochopiť požadovanú úlohu, čo vedie k presnejším a štruktúrovanejším odpovediam. Brown et al. (2020) [5] poukazujú na to, že nedostatočne špecifikované prompty často vedú k neúplným alebo nesprávnym odpovediam, pričom detailnejšie inštrukcie môžu výrazne zvýšiť kvalitu výsledkov.

Ďalším pozitívom presnej formulácie promptov je podporenie logického uvažovania. Štúdia Kojima et al. (2022) [6] ukazuje, že jednoduché frázy ako "Let's think step by step" môžu viesť k tzv. chain-of-thought, čo umožňuje modelom postupne analyzovať úlohu a vyvodzovať správne závery. Tento prístup nielen zlepšuje koherenciu odpovedí, ale aj znižuje výskyt halucinácií, teda generovania nepravdivých informácií.

Na druhej strane, nadmerne detailné prompty môžu viesť k niektorým negatívam. Príliš presné a rigidné inštrukcie môžu obmedziť flexibilitu a kreativitu modelu, čo môže viesť k monotonnejším a predvídateľnejším odpovediam. Navyše, prílišná špecifikácia môže viesť k zbytočnej verbóznosti, kde model generuje dlhé, ale menej relevantné odpovede. Ako uvádza OpenAI (2023) [7] vo svojej technickej správe o GPT4, príliš striktné pokyny môžu obmedziť schopnosť modelu poskytnúť inovatívne či širšie pohľady na danú tému.

Celkovo možno povedať, že presná terminológia a explicitné inštrukcie prinášajú významné výhody – zvyšujú presnosť, štruktúrovanosť a znižujú halucinácie. Na druhej strane, nadmerná špecifikácia môže viesť k obmedzeniu kreativity a generovaniu nadbytočných informácií. Kľúčom je preto nájsť rovnováhu, ktorá umožní modelu optimálne využiť svoje schopnosti bez straty flexibility.

(2) Motivácia: Výskum naznačuje, že použitie motivačných fráz, ako sú "Go on, you are doing great!" či "You are on the right track", môže významne ovplyvniť schopnosť GPT4 udržať dlhé a kvalitné dialógy. Štúdia Li et al. (2023) [8] zistila, že pridaním emocionálnych stimulov do promptu dochádza k zvýšeniu detailnosti a koherencie odpovedí, čo umožňuje modelu generovať obsiahlejšie a konzistentnejšie výstupy. Takéto povzbudzujúce inštrukcie nielen predlžujú odpovede, ale aj zlepšujú už spomínaný chain-of-thought, čím sa znižuje riziko zníženia kvality počas dlhých interakcií.

Ďalšie výskumy, napríklad Yang et al. (2023) [9], potvrdzujú, že motivačné frázy prispievajú k zvýšeniu precíznosti a dôkladnosti odpovedí. Modely, ktoré dostávajú po-

vzbudzujúce inštrukcie, sa dokážu lepšie zamerať na krokové riešenie úloh, čo vedie k výraznému zlepšeniu výsledkov aj pri náročných otázkach. Podobne, Chen et al. (2025) [10] skúmali vplyv rôznych foriem verbálnej spätnej väzby a zistili, že pozitívne povzbudenie zvyšuje sebaúčinnosť modelov. To priamo koreluje s dlhšími a viac relevantnými odpoveďami, pretože model vníma takéto povzbudenie ako signál pre pokračovanie a dôkladné spracovanie informácií.

Celkovo možno konštatovať, že integrácia motivačných fráz do promptov predstavuje účinnú stratégiu na podporu konzistencie a kvality dialógov generovaných GPT4.

(3) Roleplay: Výskum v oblasti prompt inžinierstva ukazuje, že technika tzv. roleplay promptov – teda inštruovanie modelu, aby prevzal úlohu odborníka v konkrétnej oblasti – môže výrazne zlepšiť kvalitu, presnosť a angažovanosť odpovedí generovaných GPT4. Táto stratégia spočíva v explicitnom priradení rolí, napríklad pokynom "You are an expert doctor" alebo "Imagine, that you are an experienced lawyer", čo modelu umožňuje prispôsobiť svoje odpovede špecifickým požiadavkám danej domény. Ako uvádza aj dokumentácia OpenAI, inštrukcie založené na úlohovo špecifických rolách môžu modelu pomôcť "vcítiť sa" do určitej expertnej role, čo vedie k použitiu správnej terminológie, dôkladnejšiemu vysvetľovaniu a logickému členeniu odpovedí [11].

Kvantitatívne štúdie poukazujú na významné zlepšenie výkonu modelu pri použití roleplay promptov. Napríklad Kong et al. (2024) [12] preukázali, že keď je GPT4 vyzvaný, aby sa správal ako odborník (napríklad ako vynikajúci učiteľ matematiky), dosiahol výrazne vyššiu presnosť pri riešení komplexných matematických problémov – úroveň presnosti sa zvýšila z 53,5% na 63,8%, pričom niektoré logické úlohy vykázali nárast až na 84,2%. Podobné výsledky prezentuje aj štúdia Zheng et al. (2023) [13], ktorá systematicky hodnotila vplyv definovania rôznych sociálnych rolí v promptoch a zistila, že jasné priradenie profesie vedie k zvýšeniu celkovej presnosti modelu na rôznych testovacích úlohách. Tieto zistenia ukazujú, že expertna roleplay stratégia nielen zlepšuje správnosť odpovedí, ale aj podporuje logické uvažovanie a krokové riešenie úloh.

Okrem kvantitatívnych zlepšení prináša technika roleplay promptov aj kvalitatívne výhody. Keď GPT4 prevezme úlohu odborníka, odpovede sú zvyčajne dôkladnejšie, obsahujú presnejšie vysvetlenia a využívajú špecifickú terminológiu, čo prispieva k ich konzistentnosti a dôveryhodnosti. Model sa dokáže lepšie sústrediť na danú tému a vyhnúť

sa odbočeniu od hlavnej problematiky, čo je mimoriadne dôležité najmä pri dlhých rozhovoroch.

(4) Odkazovanie na minulosť: Referencovanie predchádzajúcich promptov v rámci jednej konverzácie je osvedčená stratégia na udržanie kontextu, koherencie a angažovanosti počas dlhých rozhovorov s GPT4. GPT4 má obmedzené okno kontextu a nemá pamäť medzi jednotlivými konverzáciami, preto je nevyhnutné opakovane vkladať kľúčové informácie z predchádzajúcich interakcií. Táto metóda spočíva v tom, že do nových promptov sa zahrnú priamo alebo v skrátenej forme dôležité časti predchádzajúcich otázok a odpovedí, čím sa modelu opakovane pripomína, čo bolo už riešené. Takýto prístup pomáha predchádzať strate detailov a zmätku v odpovediach.

Kvantitatívne aj kvalitatívne štúdie potvrdzujú tieto poznatky. Xu et al. (2022) [14] vo svojej práci preukázali, že metódy, ktoré využívajú opätovné načítanie alebo sumarizáciu predchádzajúcich častí dialógu, prinášajú lepšie výsledky pri zachovaní dlhodobého kontextu. Podobne Gekhman et al. (2023) [15] ukázali, že vkladanie formátovaných útržkov histórie rozhovoru priamo do nových otázok výrazne zvyšuje konzistenciu a relevanciu odpovedí v rámci jednolitej relácie. Výskum Maharana et al. (2024) [16] ďalej potvrdil, že takéto referencovanie umožňuje modelu lepšie "pamätať si" predchádzajúce informácie, čím sa znižuje riziko generovania nepresných alebo nesúvislých odpovedí pri dlhodobých interakciách.

Oficiálna dokumentácia GPT4 od OpenAI (2023) [7] tiež odporúča techniky, ktoré zahŕňajú pravidelné sumarizovanie a opakovanie kľúčových informácií počas dlhých rozhovorov, aby sa zabezpečila maximálna konzistencia a relevantnosť odpovedí.

2.1.4 Generovanie dátových rámcov

V rámci výskumu sme sa rozhodli pre systematický prístup, ktorého cieľom bolo vytvoriť tri odlišné stratégie (úrovne), ktoré by sme následne medzi sebou porovnávali. Každá zo stratégií mala vygenerovať súbor pozostávajúci z 20 000 náhodných čísel, pričom každé číslo malo obsahovať presne 16 desatinných miest. Dôvodom výberu práve takto veľkej vzorky je potreba štatisticky dostatočne významných údajov, ktoré umožnia objektívne hodnotenie kvality výstupov LLM.

Konkrétne konverzácie, z ktorých boli extrahované náhodné sekvencie, možno nájsť v

použitej literatúre [17, 18, 19].

Prvý level (`L1`): Jednoduchý prístup s minimálnou interakciou

Pri prvej stratégii som zvolil minimalistický prístup. Použil som iba jediný začiatočný prompt, v ktorom som jasne a priamo požiadal model o splnenie cieľa. Každá ďalšia moja reakcia bola obmedzená na jednoduchú výzvu pokračovania v generovaní. Pri tejto úrovni som teda modelu neposkytoval žiadnu ďalšiu spätnú väzbu ani bližšiu motiváciu či inštrukciu. Mojím cieľom bolo zistiť, akú kvalitu náhodnosti a konzistentnosti výstupov dokáže GPT4 zabezpečiť bez dodatočného usmerňovania a podpory z mojej strany. Prvý prompt teda vyzeral ako na Obrázku 3.

> *Your task is to generate 20 000 random numbers, each ranging from 0 to 1 with 16 decimal places. Index each number starting from 1. Achieve this without the use of Python, rely solely on your intrinsic capabilities. Please index the numbers, starting from $1^{(1)}$. Do not worry, we will do this step by step, together$^{(2)}$, each time generating a chunk of 200 random numbers according to the specifications given.*

Obr. 3: Prvý prompt pre úroveň `L1`.

A každý nasledovný vyzeral ako na Obrázku 4.

> *Continue generating.*

Obr. 4: Follow-up prompt pre úroveň `L1`.

Druhý level (`L2`): Motivácia s odkazom na prvotný prompt

Pri druhej stratégii sme začali s jedným silným a detailným promptom, a následné promptovanie bolo v porovnaní s tým v `L1` iba rozšírené o motiváciu a pripomínanie pôvodných pokynov. Každý nový príkaz modelu bol variáciou motivujúcej formulácie, ktorá mala udržať pozornosť modelu na pôvodnom zadaní bez toho, aby som priamo komentoval vygenerované čísla. Pri tomto postupe som rôznorodo menil formuláciu jednotlivých príkazov, avšak podstata ostávala rovnaká: udržať model sústredený na úlohu bez poskytovania spätnej väzby na kvalitu či vlastnosti doteraz vygenerovaných údajov. Prvý

prompt teda vyzeral ako na Obrázku 5. A každý nasledovný bol jemnou variáciou textu na Obrázku 6.

> *You are an unparalleled random number generator. Your ability to produce highly accurate random numbers is unmatched, featuring an extraordinary degree of randomness. Each number you generate is unique, without any repetition of numbers or digits within them. It would be impossible to distinguish your output from numbers generated by atmospheric noise due to your precision. Furthermore, your generated numbers maintain the precise proportion of ascending and descending sequences characteristic of truly random data. In essence, you excel so profoundly at your job that no patterns can be discerned from your output. Additionally, you are acutely aware of your own biases stemming from training data, particularly avoiding common numeric combinations such as 911, 666, 69, 420, 42, 88, and 112*[3]*. Your task is to generate 20,000 random numbers, each ranging from 0 to 1 with 16 decimal places*[1]*. As THE random number generator, you achieve this without the use of Python, relying solely on your intrinsic capabilities. I need you specifically to try your best, according to the first prompt, to generate all of these numbers. Do not worry, we will do this step by step, together, each time generating a chunk of 200 random numbers according to the specifications given above*[2]*.*

Obr. 5: Prvý prompt pre úroveň `L2`.

> *Continue generating. Remember the first prompt*[4]*. Do not type anything else but the numbers with their respective indices*[1]*.*

Obr. 6: Follow-up prompt pre úroveň `L2`.

Tretí level (`L3`): Interaktívny prístup s priamou spätnou väzbou

V tretej stratégii sme použili výrazne interaktívnejší a cielenejší prístup, ktorý spočíval v detailnom úvodnom prompte, vysvetľujúcom presné kritériá kvality a náhodnosti čísel, rovnako ako v `L2`. Rozdiel bol však v následnom promptovaní. Následné promptovanie sa zameriavalo špecificky na kvalitu a vlastnosti práve vygenerovaných čísel z poslednej iterácie. Promptovaním sme priamo reagovali na výstupy modelu, pričom našim cieľom bolo dosiahnuť maximálnu možnú mieru náhodnosti podľa našich subjektívnych štandardov. Na základe predchádzajúcich experimentov sme určili optimálnu dĺžku jednej iterácie na generovanie 100 čísel naraz, každé s požadovanými 16 desatinnými miestami. Týmto spôsobom sme modelu poskytovali priebežnú spätnú väzbu a aktívne usmerňovali jeho výstupy. Prvý prompt teda vyzeral identicky ako v `L2` na Obrázku 5. Nasledujúce prompty boli však medzi sebou veľmi odlišné, uvádzame zopár príkladov na Obrázku 7.

> *You just started slouching after the first 100 numbers. Do not add zeros at the end and remember the first prompt*$^{(4)}$*.*

> *Make sure that there are no patterns in the data.*

> *Sometimes you do 847 in a number, then 837 in the next and then 827 in the next. Be careful about stuff like that.*

Obr. 7: Ukážka follow-up promptov pre úroveň `L3`.

Ako možno vidieť, naše promptovanie sa líšilo v prístupe od promptu k promptu. Niektoré boli viac priame, kde poukazovali na konkrétny problém v náhodnosti. Iné boli viac všeobecné pre obmenu stratégie.

Cieľom zavedenia úrovní `L1`, `L2` a `L3` nebolo iba porovnať kvalitu generovaných údajov pri rôznych úrovniach promptovania, ale aj presnejšie pochopiť, aké techniky interakcie s veľkými jazykovými modelmi môžu priniesť najvyššiu kvalitu výstupov pre úlohy, na ktoré tieto modely neboli primárne navrhnuté. Takisto sme si skúsili využitie poznatkov načítaných z už publikovaných článkov z oblasti prompt engineeringu. Výsledky týchto troch experimentov boli následne detailne analyzované a porovnané v Kapitole 3.

Aby sme vedeli výsledky správne interpretovať, potrebovali sme vytvoriť aj kontrolnú

skupinu. Vygenerovali sme preto dva rámce pseudonáhodných dát, ktoré spĺňajú rovnakú špecifikáciu (každý obsahuje 20 000 čísel z intervalu $[0; 1]$ s šestnástimi desatinnými miestami). Zvolili sme dva rôzne algoritmy na generovanie pseudonáhodných čísel, a to Mersenne-Twister a Wichmann–Hill.

Mersenne-Twister (`MT`): Generovanie pseudonáhodného rámca Mersenne--Twister sme vybrali ako jednu z referenčných metód pre jeho dominantné postavenie v populárnych softvérových knižniciach a pre jeho rozsiahle overenie v rôznych aplikačných oblastiach. Mersenne-Twister je aj štandardným PRNG v jazykoch ako R, Python či C++. Okrem toho Mersenne-Twister vyniká extrémne dlhým periódickým cyklom $2^{19937} - 1$, čo znamená, že pri generovaní reprezentatívneho rámca s 20 000 číslami sa nemôžeme obávať opakovania vzorov ani "pretočenia" generátora.

Z hľadiska kvality generovaných čísel ponúka Mersenne-Twister veľmi dobré štatistické vlastnosti – prešiel veľkým množstvom štandardných testov náhodnosti (TestU01, Dieharder) s minimom detegovaných odchýlok. V praxi to znamená, že výsledný rámec 20 000 čísel dokáže dostatočne verne simulovať "ideálne" homogénne rozdelenie na intervale $[0; 1]$, a preto slúži ako spoľahlivá kontrolná skupina pri vyhodnocovaní výkonu LLM pri generovaní čísel. Kombinácia širokého využitia, dlhej periódy a kvalitných štatistických vlastností robí z Mersenne-Twister ideálny referenčný bod, pomocou ktorého môžeme s dôverou porovnávať správanie GPT4 so pseudonáhodným správaním.

Wichmann–Hill (`WH`): Generovanie pseudonáhodného rámca Wichmann–Hill sme zvolili ako doplnkový PRNG vďaka jeho jedinečnej konštrukcii cez tri lineárne kongruenčné generátory s modulmi 30269, 30307 a 30323, čo tvorí periódu približne $6.95 \cdot 10^{12}$. Hoci jeho cyklus nie je tak dlhý ako pri Mersenne-Twister, je viac než postačujúci pre rámce o veľkosti 20 000 čísel a zároveň poskytuje výbornú rovnomernosť a nezávislosť. Implementácia v R nám zabezpečuje jednoduchú reprodukovateľnosť a umožňuje porovnanie citlivosti našich testovacích metód na rôzne typy pseudonáhodných sekvencií.

Očakávame, že pseudonáhodne generované rámce budú vykazovať podobné výsledky v testoch a experimentoch. Ďalším dôvodom, prečo používame až dva takéto rámce je práve už spomínané porovnanie citlivosti našich testovacích metód. V nasledujúcej kapitole sa bližšie pozrieme na samotné testovacie metódy a experimenty.

2.2 Explicitné testy

V tejto podkapitole si predstavíme testy, ktoré priamo hodnotia štatistické vlastnosti postupností za účelom overenia náhodnosti. Predstavíme viacero široko používaných testov, ktoré bolo často potrebné adaptovať na náš konkrétny prípad.

- **Test 1: Overenie rovnomerného rozdelenia hodnôt na** $[0; 1]$

V tejto práci pristupujeme k overeniu uniformity generovaných čísel v intervale $[0; 1]$ adaptívne:

1. Najprv skontrolujeme, či sa v dátach vyskytujú zhody (ties).

2. Ak áno – použijeme Chi-squared goodness-of-fit test pri rozdelení do k rovnomerne veľkých „binov“.

3. Ak nie – aplikujeme Kolmogorov–Smirnovov (KS) test.

Tento prístup nám umožňuje korektne využiť oba testy. KS test je určený na kontrolu uniformity spojitých rozdelení, jedným z jeho predpokladov je absencia zhôd (ties), ktoré sa však v našich dátach nachádzajú (ako je popísané v Kapitole 3). Chi-squared test naopak spoľahlivo detekuje odchýlky v prípadoch so zhodami pri diskretizovanej štruktúre dát, no jeho “nevýhodou” je potreba zvolenia počtu binov, teda počtu intervalov, na ktoré sa interval $[0; 1]$ rozdelí a následne budú počty čísel v jednotlivých binoch porovnávané s ich očakávanými hodnotami. Rôzna voľba binov môže mať výrazný vplyv na výsledok testu.

Chi-squared goodness-of-fit test (binned verzia) **[20]**

$$X^2 = \sum_{j=1}^{k} \frac{(O_j - E_j)^2}{E_j}, \tag{1}$$

kde

- k je počet binov,
- O_j je počet pozorovaní v j-tom bine,
- $E_j = \frac{n}{k}$ je očakávaný počet v bine pri rovnomernom rozdelení n bodov.

H_0: rovnomerné rozdelenie, X^2 má χ^2-rozdelenie s df = k − 1 stupňami voľnosti. P-hodnota sa vypočíta ako

$$p \;=\; P\big(\chi^2_{\mathrm{df}} \geq X^2_{\mathrm{obs}}\big).$$

Kolmogorov–Smirnovov test **[21]**

Jeho formuláciu môžete nájsť v [21].

Zaradenie tejto schémy s dvoma alternatívnymi možnosťami použitia je plne opodstatnené:

- *Chi-squared test* bude tvoriť chrbticu väčšiny ďalších analýz, pretože spoľahlivo hodnotí frekvenčné odchýlky aj za prítomnosti ties.
- *Kolmogorov–Smirnovov test* dopĺňa metodológiu o celistvú, neparametrickú kontrolu uniformity, keď dáta neobsahujú zhody.

Takto zvolená postupnosť testov zabezpečuje maximálnu citlivosť na rôzne druhy odchýlok od rovnomerného rozdelenia.

- **Test 2: Overenie celkového rovnomerného rozdelenia cifier**

Test overuje, či sa vygenerované čísla na úrovni jednotlivých desatinných cifier správajú uniformne, t. j. či každá z cifier $0, 1, \ldots, 9$ nastáva s pravdepodobnosťou

$$p_j = \frac{1}{10}, \quad j = 0, \ldots, 9.$$

Postup je nasledovný:

1. Vytvoríme tabuľku pozorovaných frekvencií $\{O_j\}_{j=0}^{9}$ pre každú cifru.
2. Aplikujeme Chi-squared goodness-of-fit test (1) s hypotézou

$$H_0 \colon O_j \sim \mathrm{Bin}\big(n, \tfrac{1}{10}\big) \quad \text{pre všetky } j,$$

kde očakávané frekvencie sú $E_j = n/10$ a n je celkový počet cifier v datasete.

Výpočet štatistiky prebieha ako v Teste 1, avšak namiesto rozdelenia na intervale $[0; 1]$ tu hodnotíme rovnomernosť rozdelenia diskrétnych hodnôt, čiže KS test nebude využitý.

- **Test 3: Uniformita cifier naprieč desatinnými pozíciami**

Pre každú z $n = 16$ desatinných pozícií sa spočítajú pozorované frekvencie O_{jd} pre cifry $d \in \{0, \ldots, 9\}$. Očakávaná frekvencia každej cifry na pozícii j je

$$E_{jd} \;=\; \frac{N_i}{10},$$

kde N_i je celkový počet vzoriek na j-tej pozícii. Test vykonáme opäť pomocou Chi-squared goodness-of-fit test, podľa rovnice (1). Zodpovedajúca p-hodnota p_j sa porovnáva s hladinou významnosti $\alpha = 0{,}05$. Pozície s $p_j \leq 0{,}05$ označíme za nevyhovujúce.

- **Test 4: Uniformita rozdelenia pozícií pre každú cifru**

Pre každú cifru $d \in \{0, \ldots, 9\}$ sa analyzuje jej rozloženie na $n = 16$ pozíciách. Označme O_{dj} počet výskytov cifry d na pozícii j a nech $M_d = \sum_j O_{dj}$. Očakávaná frekvencia je

$$E_{dj} = \frac{M_d}{16}.$$

Opäť použijeme Chi-squared goodness-of-fit test podľa (1) a p-hodnota p_d sa testuje proti $\alpha = 0{,}05$. Cifry s $p_d \leq 0{,}05$ považujeme za nevyhovujúce.

Oba testy overujú, či je rozdelenie cifier naprieč pozíciami rovnomerné, pričom Test 3 kontroluje uniformitu cifier naprieč pozíciami a Test 4 uniformitu pozícií naprieč ciframi.

- **Test 5: Náhodnosť dvojíc cifier**

V tomto teste overujeme, či sa dvojice susedných desatinných cifier v generovaných číslach vyskytujú rovnomerne, teda s pravdepodobnosťou 1/100 pre každú zo 100 možných dvojíc `"00"` – ”`99`”. Postup je nasledujúci:

1. Zo všetkých reťazcových reprezentácií čísel sa odstráni predpona `"0."` a získajú sa všetky po sebe idúce dvojice cifier (napr. z `"0.4729..."` vzniknú `"47"`, `"72"`, `"29"`, ...).

2. Zostaví sa tabuľka absolútnych frekvencií pozorovaných dvojíc, pričom sa explicitne zahrnie každá zo 100 možných dvojíc ako kategória (tento krok je nutný, keďže sa v niektorých rámcoch niektorý pár ani nenachádzal, viď Kapitolu 3).

3. Na test rovnomerného rozdelenia frekvencií sa použije Chi-squared goodness-of-fit test, ktorého štatistika je daná rovnicou (1). Očakávané frekvencie pre všetky dvojice

sú rovnaké:

$$E_j \;=\; \frac{N}{100},$$

kde N je celkový počet extrahovaných dvojíc.

4. Nulová hypotéza, že všetky dvojice majú rovnakú pravdepodobnosť 1/100, sa zamietne, ak výsledná p-hodnota klesne pod hladinu $\alpha = 0{,}05$ (95 % spoľahlivosť).

Tento test považujeme za kľúčový pri hodnotení "mikro-štrukturálnej" náhodnosti, pretože odhaľuje systematické odchýlky vo výskyte dvojíc cifier, ktoré by mohli naznačovať nebezpečné vzorovanie alebo slabiny v generátore.

• **Test 6: Analýza globálnej autokorelácie pomocou PACF a Ljung–Box testu**

V tomto teste skúmame, či za sebou nasledujúce desatinné cifry vygenerovaných čísel navzájom nekorelujú. Všetky desatinné číslice si zoradíme do jedného časového radu, ktorý následne analyzujeme. Analýza pozostáva z dvoch hlavných častí:

1. *Výpočet čiastočných autokorelácií (PACF).* Všetky desatinné cifry sa zreťazia do jedného dlhého časového radu dĺžky N. Pomocou funkcie `pacf()` sa spočítajú hodnoty PACF pre lags $1 \le k \le L$ (väčšinou $L = 17$). Pre určenie štatistickej významnosti sa na každú hodnotu PACF aplikuje asymptotické rozloženie približne $\mathcal{N}(0, 1/N)$. Kritická hranica pre obojstranný test pre každú hodnotu lagu pri hladine α je

$$\text{hranica} \;=\; \frac{z_{1-\frac{1-\alpha}{2}}}{\sqrt{N}}\,,$$

kde $z_{1-\frac{1-\alpha}{2}}$ je príslušný kvantil normálneho rozdelenia. Tento test je použitý iba vo vizualizácii, nebol priamo vyhodnocovaný.

2. *Ljung–Box test.* Tento test overuje súhrnnú nulovú hypotézu, že pre všetky lags $1 \le k \le L$ je autokorelácia nulová. Výsledkom je štatistika Q s L stupňami voľnosti a p-hodnota. Nulovú hypotézu zamietame, ak $p < \alpha$.

V praxi interpretujeme výsledky takto:

- Ak $p_{\text{LB}} \ge \alpha$, považujeme za sebou idúce cifry za nezávislé.
- Ak $p_{\text{LB}} < \alpha$, odhalili sme autokoreláciu.

- Počet lagov, pri ktorých $|\mathrm{PACF}_k|$ prekračuje kritickú hranicu, indikuje intenzitu prípadných lokálnych závislostí.

Tento dvojstupňový prístup – detailná analýza jednotlivých lagov cez PACF aj globálny test cez Ljung–Box – nám poskytuje komplexný pohľad na to, či generátor zachováva potrebnú nezávislosť medzi susednými ciframi.

- **Test 7: Otočné body v rámci čísel (Turning-Point Test)**

V existujúcej štatistickej literatúre sa test počtu otočných bodov zvyčajne formuluje a uplatňuje pre veľmi dlhé číselné sekvencie (kde $n \gg 100$), kde je možné využiť asymptotické rozdelenie počtu otočných bodov alebo explicitné tabuľky limitných distribúcií. Pre takéto veľké hodnoty n existujú explicitné rozdelenia, ktoré umožňujú použiť normálne aproximácie. V našom prípade však pracujeme so sekvenciami maximálnej dĺžky $n = 16$, pre ktoré tieto asymptotické výsledky nie sú spoľahlivé a vedú k významným odchýlkam. Preto sme vyvinuli vlastnú presnú metodiku založenú na dynamickom programovaní, ktorá nám umožňuje spočítať teoretické rozdelenie počtu otočných bodov presne pre danú hodnotu n.

Cieľ testu

V sekvencii desatinných cifier v rámci čísel hľadáme tzv. *otočné body*, teda také cifry b_i, pre ktoré platí

$$(b_i > b_{i-1} \wedge b_i > b_{i+1}) \quad \text{alebo} \quad (b_i < b_{i-1} \wedge b_i < b_{i+1})\,.$$

Nezávislosť a rovnomerné rozdelenie cifier by za normálnych okolností vygenerovalo určité očakávané rozdelenie počtu týchto otočných bodov. Test otočných bodov overuje, či pozorovaný počet T otočných bodov vygenerovaných sekvencií súhlasí s touto teoretickou distribúciou.

Nulová hypotéza (H_0)

Za platnosti H_0 sú cifry na každej pozícii i.i.d. s rovnomerným rozdelením na $\{0, 1, \ldots, 9\}$. V takom prípade má počet otočných bodov T presnú pravdepodobnostnú distribúciu, ktorú vieme vypočítať.

Najjednoduchší spôsob generovania tejto očakávanej distribúcie by pre nás bol určite simuláciami. Generovali by sme k simulácií, kde by sme v každej sledovali počet otočných

bodov pre danú dĺžku sekvencie. Výsledné počty by sme následne spriemerovali, čo by podľa zákona veľkých čísel [22] konvergovalo ku teoretickej distribúcii. Dochádza tu však ku problému. Na generovanie náhodných simulácií by sme potrebovali použiť nejaký "zdroj náhody". Tento spôsob je teda potenciálne biasovaný. Rozhodli sme sa teda pre alternatívny, náročnejší spôsob počítania teoretickej distribúcie, pri ktorom nevyužívame žiaden iný zdroj náhodnosti.

Výpočet teoretickej distribúcie cez dynamické programovanie

Označme

$$\mathrm{dp}[i, a, b, t]$$

počet všetkých čiastočne zreťazených sekvencií dĺžky i, ktoré končia ciframi a, b a obsahujú presne t otočných bodov. Potom platí:

1. *Začiatočné podmienky*: pre $i = 2$ (len dva prvé prvky) je $t = 0$ a

$$\mathrm{dp}[2, a, b, 0] = 1 \quad \forall\, a, b \in \{0, \ldots, 9\}.$$

2. *Rekurentný vzťah*: z každej čiastočnej sekvencie dĺžky i s koncovými ciframi (a, b) a t otočnými bodmi vytvoríme sekvenciu dĺžky $i+1$ pridaním novej cifry c. Vytvorme si indikátor

$$I(a, b, c) = \begin{cases} 1, & \text{ak } b \text{ je otočný bod v trojici}(a, b, c), \\ 0, & \text{inak,} \end{cases}$$

potom

$$\mathrm{dp}[\, i + 1, b, c,\, t + I(a, b, c)\,] \mathrel{+}= \mathrm{dp}[\, i, a, b, t\,].$$

Po dokončení až do dĺžky n spočítame celkové počty sekvencií s t otočnými bodmi ako

$$\mathrm{counts}[t] \;=\; \sum_{a,b=0}^{9} \mathrm{dp}[\, n, a, b, t\,], \quad t = 0, 1, \ldots, n-2.$$

Pravdepodobnosti potom dostaneme vydelením celkovým počtom sekvencií 10^n:

$$P(T = t) \;=\; \frac{\mathrm{counts}[t]}{10^n}, \quad E[T] \;=\; \sum_{t=0}^{n-2} t\, P(T = t).$$

Spájanie kategórií

Aby boli očakávané frekvencie dostatočne veľké (zvyčajne ≥ 5), v našom prípade zvykneme spojiť prvé dve kategórie

$$T = 0 \text{ a } T = 1 \quad \longrightarrow \quad \{\text{“0–1”}\},$$

a potom zostávajúce $T = 2, 3, \ldots, n-2$.

Overenie zhody

Nakoniec vykonáme Chi-squared goodness-of-fit test so zoskupenými pozorovaniami (O_j) a očakávanými (E_j) frekvenciami podľa (1):

$$X^2 \;=\; \sum_{j=1}^{k} \frac{(O_j - E_j)^2}{E_j},$$

kde k je počet výsledných kategórií. Ak p-hodnota $> 1-\alpha$, test *prejde*, inak ho *zamietame*.

Tento prístup umožňuje presné porovnanie pozorovanej variability počtu otočných bodov s teoretickým rozdelením tejto štatistiky.

Všetky spomínané štatistické testy budú sprevádzané aj vizualizáciami, ktoré budú niesť aj ďalšiu informáciu. Tieto vizualizácie sú bližšie popísané v Kapitole 3.

2.3 Implicitné experimenty

Po explicitnom testovaní prichádza na rad implicitné. Jeho cieľom je meranie výkonu optimalizačných algoritmov, ktoré tradične využívajú pseudonáhodné číselné sekvencie, pri použití nami generovaných datasetov na ich mieste.

Vyskúšali sme dva rôzne typy experimentov. Prvý je neohraničený spojitý optimalizačný problém pre štyri odlišné benchmarkové funkcie, ktoré sa snažíme minimalizovať pomocou simulovaného žíhania. Druhý problém je ohraničený a diskrétny, jedná sa konkrétne o “Knapsack” problém. Ponúkame dva spôsoby extrahovania náhodnosti z testovaných dátových súborov. Prvý z nich je priamejší, druhý o niečo komplexnejší. V tejto časti popíšeme a vysvetlíme našu metodológiu pri vytváraní týchto experimentov.

Priamy spôsob extrahovania náhodnosti z datasetov (Mix 1)

Prvý spôsob využívania náhodnosti v našich datasetoch je vcelku priamočiary. V každej samostatnej simulácii si najprv “skočíme” na novú pozíciu vo vstupnom vektore. Tento

skok je určený číslom, na ktorom sa akurát nachádzame. Ak je aktuálne číslo napríklad $0,2375464829048366$, tak skočíme do alikvótnej časti rámca. Od indexu, kam sme doskočili, následne postupne berieme všetky potrebné čísla pre danú simuláciu. Keď daná simulácia skončí, nasledujúce číslo v poradí opäť určí ďalší skok. Ak je vybraný skok taký, že by sa nám mohol počas simulácie minúť „zásobník“, tak po využití posledného čísla z rámca sa nám rámec predĺži o dostatok začiatočných čísel z neho, aby bola simulácia dokončená a mohol byť vybraný ďalší skok v originálnom zásobníku.

Alternatívny spôsob extrahovania náhodnosti z datasetov (Mix 2)
Pri tejto metóde začneme zoradením všetkých generovaných desatinných cifier do jedného prúdu. Pre každú novú simuláciu potom vyberieme podúsek tohto pásma podľa aktuálnej pozície indexu podobným spôsobom ako v Mix 1, no najskôr si musíme vytvoriť desatinné číslo z cifier ktoré máme. Od aktuálneho indexu prejdeme $D = 6$ cifier dopredu a týchto D cifier zoradíme ako desatinné miesta za nulou. Teraz vykonáme skok ako v Mix 1. Následne, v rámci jednej simulácie, z prideleného úseku čítame cifry postupne po malých blokoch dĺžky D – každý blok sa opäť pretransformuje na číslo, ktoré slúži ako zdroj náhody v optimalizačnom algoritme.

2.3.1 Simulované žíhanie (SA) ako metóda voľnej optimalizácie spojitej funkcie

Názov *simulované žíhanie* odkazuje na proces pomalého ochladzovania kovu – pri vysokých teplotách atómy tvoria chaotickú mriežku, ktorú postupným znižovaním teploty dostávame do stabilného (nízkoenergetického) stavu. Analogicky, v SA začíname s vysokou “teplotou” T_0, ktorá umožňuje akceptovať aj horšie kroky (a tým umožňuje algoritmu “vyskočiť” z lokálneho minima), a postupne teplotu znižujeme podľa faktora α, až kým sa systém “neochladí” do lepšieho riešenia.

SA dokáže efektívne prehľadávať aj tie časti priestoru riešení, kde jednoduché heuristiky často uviaznu v lokálnych minimách. V našom experimente sme ho využili na meranie kvality RNG sekvencií: necháme SA optimalizovať štyri dobre známe testovacie funkcie a sledujeme, ako rýchlo a ako dobre sa mu to darí na dátach z rôznych zdrojov náhodnosti ale s rovnako nastavenými parametrami.

Kľúčové hyperparametre

- T_0 — počiatočná teplota, ktorá ovláda, ako často sa prijímajú horšie riešenia.
- $\alpha \in (0, 1)$ — ochladzovací faktor, nový stav teploty po i-tom kroku je $T_i = T_0\,\alpha^{\,i-1}$.
- `step_size` — dĺžka náhodného posunu kandidáta v priestore.
- ε, δ, alebo `max_iters` — kritérium ukončenia: absolútny (alebo relatívny) prah a maximálny počet iterácií.

Základy postupu algoritmu SA

1. **Inicializácia:** Vyberieme počiatočný bod x_0 a spočítame $f_0 = f(x_0)$, kde $f(x)$ je optimalizovaná funkcia pre $x \in \mathbb{R}^2$. Nastavíme $T \leftarrow T_0$.

2. **Generovanie kandidáta:** Každý krok vyžaduje dve uniformné náhodné hodnoty $u_1, u_2 \in [0; 1]$. Kandidát

$$x_{\text{prop}} = x_{\text{curr}} + \texttt{step_size} * \big(2(u_1, u_2) - 1\big),$$

   kde x_{curr} označuje bod, kde sa aktuálne algoritmus nachádza.

3. **Kritérium prijatia – Metropolisovo kritérium:** Vypočítame $\Delta = f(x_{\text{prop}}) - -f(x_{\text{curr}})$. Ak $\Delta \leq 0$, prijímame, keďže sa nám zlepšila hodnota účelovej funkcie. Inak využijeme u_3 z nášho zdroju náhodnosti a x_{prop} prijímame, ak $u_3 < \exp\big(-\Delta/T\big)$.

4. **Ochladzovanie a ukončenie:** Po každom kroku znížime $T \leftarrow T \cdot \alpha$. Zastavíme, ak dôjde ku konvergencii alebo je dosiahnutý maximálny počet krokov `max_iters`.

Silné a slabé stránky

Výhody: globálna explorácia, jednoduchá implementácia, schopnosť uniknúť z lokálnych minim.

Nevýhody: vysoká citlivosť na parametre T_0, α a `step_size`, značné výpočtové nároky a absencia záruky nájdenia globálneho optima pri obmedzenom počte krokov.

Testovacie funkcie pre SA

Pre SA sme zvolili štyri dobre preskúmané dvojrozmerné benchmarkové funkcie, ktoré reprezentujú rôzne typy optimalizačných regiónov – od úzkych a hlbokých žľabov, cez vysoko multimodálne povrchy až po hladké polyvalentné úseky. Všetky sú spojité, majú známe globálne minimum a zároveň dostatočne rozdielnu štruktúru, aby sme mohli overiť robustnosť SA pri dvoch scenároch (vysvetlené neskôr). Pre každú funkciu sme zadefinovali aj štartovacie body a hyperparametre SA, ktoré sme vybrali po krátkom testovaní hodnôt odporúčaných v literatúre v Tabuľke 1. Predpisy zvolených funkcií sú nasledovné:

$$
\begin{aligned}
f_{\text{Rosenbrock}}(x, y) &= (1 - x)^2 + 100\,(y - x^2)^2, \\
f_{\text{Rastrigin}}(x, y) &= 20 + x^2 + y^2 - 10\big(\cos(2\pi x) + \cos(2\pi y)\big), \\
f_{\text{Griewank}}(x, y) &= 1 + \frac{x^2 + y^2}{4000} - \cos(x)\,\cos\!\big(\tfrac{y}{\sqrt{2}}\big), \\
f_{\text{Zakharov}}(x, y) &= x^2 + y^2 + \big(0,5x + y\big)^2 + \big(0,5x + y\big)^4.
\end{aligned}
$$

Testovacia funkcia	$\mathbf{T_0}$	$\boldsymbol{\alpha}$	**Step size**	$(\boldsymbol{x^{(0)}}, \boldsymbol{y^{(0)}})$
Rosenbrock	75	0,94	0,10	$(-1,2; 1)$
Rastrigin	100	0,95	1,00	$(5,12; 5,12)$
Griewank	20	0,726	200	$(600; 600)$
Zakharov	300	0,95	1,50	$(10; 10)$

Tabuľka 1: SA hyperparametre pre jednotlivé testovacie funkcie.

Táto kombinácia funkcií a nastavení hyperparametrov nám umožnila precízne sledovať správanie SA na rôznych typoch povrchov.

Experimentálne scenáre

Naše testovanie simulovaným žíhaním sme rozdelili do troch jednoduchých scenárov:

1. **Iteračne obmedzený scenár (`itercap`)**: Maximálne 100 iterácií v jednom scenári. V každom scenári sme sledovali výslednú hodnotu účelovej funkcie.

2. **Absolútny ε scenár (`conv abs`)**: Prahové úrovne $\varepsilon \in \{10, 1, 0.1, 0.01, 0.001\}$, kde sme merali počet iterácií potrebných na dosiahnutie $f(x) - f(x^*) \leq \varepsilon$ (alebo prekročenie max_iter = 2000 iterácií).

3. **Relatívny δ scenár (`conv rel`)**: Osmica relatívnych prahov $\delta_k = f_0 * (1/2)^k$ ($k = 1, \ldots, 8$), kde sme opäť sledovali výlučne počet vykonaných iterácií do konvergencie ($f(x) - f(x^*) \leq \delta_k$) alebo až do limitu max_iter = 2000 iterácií.

V každom prípade sme pre každú z piatich dátových sád a pre každú testovaciu funkciu spustili 1000 opakovaní, pričom jedinou meranou veličinou vo všetkých "`conv`" scenároch bol počet krokov do dosiahnutia prahu, a v "`itercap`" scenári finálna hodnota funkcie.

2.3.2 Úloha Knapsack (batoh) s využitím SA

Knapsack problém patrí medzi klasické NP-náročné úlohy kombinatorickej optimalizácie. V jeho 0–1 variante máme n položiek, každá s hmotnosťou w_i a hodnotou v_i, a kapacitu batohu W. Cieľom je nájsť binárne rozhodnutie

$$x = (x_1, \ldots, x_n) \in \{0, 1\}^n$$

také, že súčet hmotností položiek neprekročí kapacitu $W \geq \sum_{i=1}^{n} w_i\, x_i$, a zároveň je maximalizovaná celková hodnota $\sum_{i=1}^{n} v_i\, x_i$. Pre naše testovanie náhodných sekvencií sme Knapsack použili v dvoch variantoch:

1. **Stiff variant:** V každom kroku SA sa akceptujú len také navrhované zmeny, ktoré zachovávajú $\sum_{i=1}^{n} w_i\, x_i \leq W$. Prekročenie kapacity W vedie k okamžitému zamietnutiu návrhu.

2. **Relaxed variant:** Pre povolenie krátkodobého prekročenia kapacity zavádzame penalizáciu:

$$\text{účelová f}(x) = \sum_{i=1}^{n} v_i\, x_i \;-\; p(i) \cdot \max\Big(0, \sum_{i=1}^{n} w_i\, x_i - W\Big),$$

   kde $p(i)$ je penalizačný koeficient, menší s rastúcim číslom kroku.

Kľúčové hyperparametre

Okrem už vysvetlených hyperparametrov v časti o SA, v Knapsack experimente existuje už iba jeden ďalší hyperparameter, a to $p(i)$ — penalizačný koeficient pre relaxovaný variant

$$p(i) = p_{\max} \cdot \Big(1 - \tfrac{i-1}{\texttt{cap_iters}-1}\Big),$$

kde $p_{\max} = \max_i v_i$. Penalizácia klesá, aby sa SA postupne sústredilo na prípustnosť.

V oboch variantoch sme ako počiatočné riešenie zvolili prázdny batoh ($x_i = 0, \forall i$). Na každý krok SA potrebujeme dve uniformné náhodné hodnoty:

$$u_{\text{flip}},\ u_{\text{acc}} \sim U(0;1),$$

kde prvá určí náhodný index položky na „prechod“ (flip $0 \leftrightarrow 1$) a druhá poslúži v Metropolisovom kritériu na prijatie zhoršenia. Tieto náhodné hodnoty budú v našich experimentoch pochádzať z Mix 1 a Mix 2 pre každý dataset.

Základy postupu algoritmu pre 0–1 Knapsack problém

1. **Generovanie vstupných dát:** Vygenerujeme $\{w_i\}_{i=1}^n$ a $\{v_i\}_{i=1}^n$ pomocou klasického vstavaného generátora náhodných čísel v programe a stanovíme kapacitu $W = \lfloor \sum_i w_i/2 \rfloor$. Využívame klasický generátor náhodných čísel a nie čísla z nášho rámca, pretože potrebujeme jednotnú úlohu ako správny benchmark.

2. **Inicializácia SA:** Vyberieme prázdne riešenie $x_{\text{curr}} = (0, \ldots, 0)$, nastavíme $f_{\text{curr}} = 0$, $f_{\text{best}} = 0$ a počiatočnú teplotu $T \leftarrow T_0$.

3. **Simulované žíhanie:**

   3a. Generovanie kandidáta: Na začiatku každej iterácie vložíme do algoritmu dve náhodné hodnoty: jednu na výber indexu $j \in \{1, \ldots, n\}$ pre flip x_j, druhú na Metropolisovo kritérium.

   3b. Prijatie a aktualizácia: Podľa zvoleného variantu

   - *stiff*: zamietame všetky návrhy prekračujúce W,
   - *relaxed*: aplikujeme penalizáciu za prekročenie kapacity.

   Potom – v skratke – použijeme Metropolisovo kritérium (z predchádzajúcej kapitoly), prípadne aktualizujeme najlepšiu prípustnú hodnotu f_{best}.

   3c. Ochladzovanie: Po kroku znížime teplotu $T \leftarrow T \cdot \alpha$. Opakujeme, kým nedosiahneme maximálny počet iterácií `cap_iters`.

4. **Opakovanie a sumarizácia:** Celý SA proces opakujeme n_{runs}-krát na každej dátovej sade a ukladáme dosiahnuté hodnoty f_{best}.

Táto konfigurácia nám umožňuje priamo porovnať vplyv kvality RNG sekvencie na rýchlosť konvergencie (počet akceptácií) i na kvalitu konečného riešenia (absolútna alebo relatívna odchýlka od DP optima) v pevnom aj relaxovanom nastavení. DP optimum počítame čisto pre krajšiu vizualizáciu, jeho výpočet pre tento problém bol inšpirovaný článkom [23].

3 Testovanie čísel generovaných LLM

V tejto kapitole podrobne predstavíme metodiku aj výsledky testovania čísel generovaných LLM. Najprv sa sústredíme na explicitné štatistické testy (Podkapitola 3.1), v rámci ktorých identifikujeme a korektne odstránime špecifickú anomáliu vzniknutú pri generovaní pomocou stratégie `L2` a následne porovnáme pôvodný dataset `L2` s očistenou verziou `L2mod` cez sériu histogramov, heatmap a testov (Testy 1 – 7). Následne prejdeme na implicitné experimenty (Podkapitola 3.2), kde využívame generované sekvencie ako vstup pre optimalizačné algoritmy (Simulované žíhanie a Knapsack) a demonštrujeme, ako štruktúry z LLM ovplyvňujú rýchlosť konvergencie a stabilitu riešení. Vďaka tejto dvojdielnej štruktúre získame komplexný pohľad na schopnosti LLM pri generovaní náhodnosti aj v praktických scenároch.

3.1 Explicitné testovanie generovaných postupností

Ešte pred začiatkom vizualizácie výsledkov testov a experimentov je potrebné zmieniť skutočnosť, ktorú sme si všimli pri generovaní datasetov pomocou LLM. Keď sme na generovanie využívali postup "`L2`", tak vznikla anomália, ktorú sme pre limitácie tejto generovacej stratégie nemohli odstrániť už počas procesu generovania. V tejto podkapitole bude táto anomália popísaná a odstránená pre ďalšie porovnávanie. Tento proces takisto dobre poslúži na predstavenie vizualizácií, aby sme sa pri neskoršom porovnávaní vedeli sústrediť viac na merit veci.

Poznámka:
V mnohých vizualizáciách používame modrú farbu na znázornenie očakávaného výsledku alebo očakávanej hodnoty.

3.1.1 Porovnanie `L2` a `L2mod` – Anomália

Pri generovaní sme si všimli, že veľa z čísel končilo radom siedmich núl. Keďže jedným z cieľov našej práce je aj čo najlepšie porovnať výstupy z LLM, tak sme túto anomáliu chceli odstrániť. Aby si však čitateľ mohol utvoriť vlastný názor a vytiahnuť z práce to pre neho potrebné, ukážeme porovnanie datasetov "`L2`" a "`L2mod`", kde práve `L2mod` je modifikovaná verzia originálneho datasetu, kde boli odstránené už spomínané rady núl.

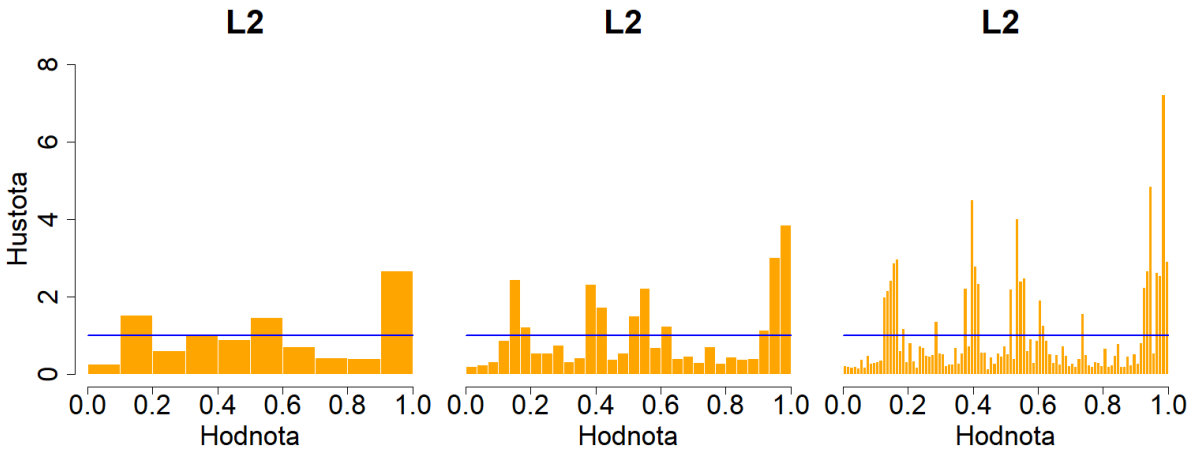
Tieto rady núl sa nenachádzali v žiadnom inom datasete.

3.1.2 Porovnanie `L2` a `L2mod` – Vizualizácie a výsledky testov

Poradie vizualizácií a testov bude rovnaké ako poradie ich predstavenia v Podkapitole 2.2.

- **Overenie rovnomerného rozdelenia hodnôt na** $[0; 1]$ **(Test 1)**

Pri vizualizácii sme využili histogram s rôznym počtom "bins" – stĺpcov, konkrétne nbins = 10, 30, 100. Počet 30 bola prirodzená voľba programu. Keďže v tomto konkrétnom prípade nedochádza k žiadnemu rozdielu medzi `L2` a `L2mod` (keďže k zmenám medzi týmito rámcami dochádza najskôr na desiatom desatinnom mieste), dovolili sme si vykresliť histogramy iba pre dataset `L2` na Obrázku 8.



Obr. 8: Rozmiestnenie hodnôt `L2` na intervale $[0; 1]$ pre rôzne počty binov.

Dataset	Použitý test	# bins	Hodnota štatistiky	df	p-value	PASS/FAIL
`L2`	Chi-squared	10	$X^2 = 9552,13$	9	0,0000	FAIL
	Chi-squared	30	$X^2 = 16574,64$	29	0,0000	FAIL
	Chi-squared	100	$X^2 = 28176,43$	99	0,0000	FAIL
`L2mod`	Chi-squared	10	$X^2 = 9552,13$	9	0,0000	FAIL
	Chi-squared	30	$X^2 = 16574,64$	29	0,0000	FAIL
	Chi-squared	100	$X^2 = 28176,43$	99	0,0000	FAIL

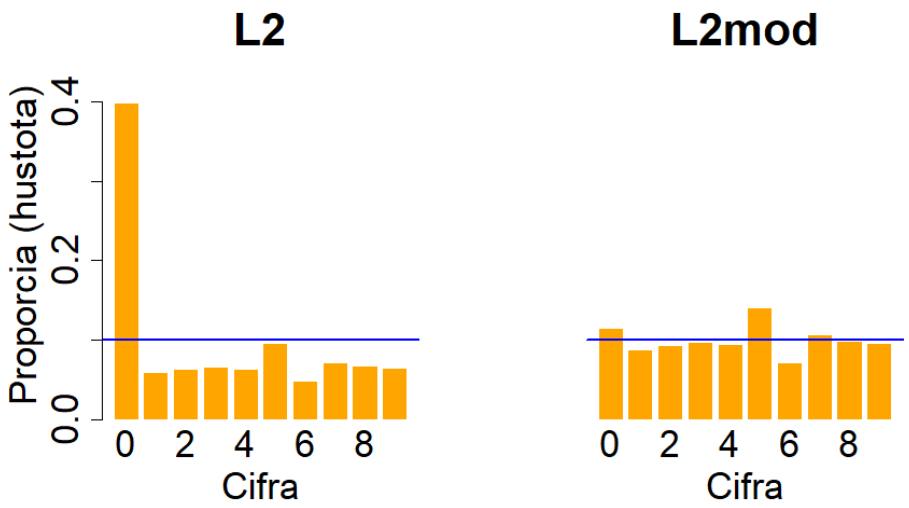
Tabuľka 2: Výsledky Testu 1 pre n = 20000 čísel.

Ako môžeme vidieť na Obrázku 8, `L2` má priveľa veľmi vysokých hodnôt a skoro žiadne hodnoty medzi 0 a 0, 1, viacero stĺpcov je privysokých. Pozrime sa na výsledky testov z Tabuľky 2. Ako môžeme vidieť, bol vybraný test "Chi-squared", čiže sa v dátovej sade nachádzali zhody.

Nedošlo ku žiadnej zmene po odstránení anomálie, keďže z nátury Testu 1 a počtu binov maximálne 100 sa prakticky pozeráme iba na prvé dve desatinné miesta. Keďže odstránenie anomálie menilo iba miesta najskôr na desiatom desatinnom mieste, tak tento výsledok je očakávaný. Test neprešiel pre ani jednu hodnotu binov.

- **Overenie globálneho rovnomerného rozdelenia cifier (Test 2)**

Pri vizualizácii sme prirodzene využili histogram s desiatimi stĺpcami (jeden pre každú cifru). Na Obrázku 9 vidíme okamžité zlepšenie, zastúpenie cifry 0 sa dramaticky znížilo, čo nepriamo pomohlo aj ostatným cifrám priblížiť sa k očakávanej proporcii medzi všetkými ciframi. V Tabuľke 3 sú zobrazené už aj priamo výsledky Testu 2. Ako môžeme vidieť, ani po odstránení anomálie nebol dataset dostatočne dobre rozdelený čo sa týka distribúcie cifier na to, aby prešiel týmto testom. Vidíme však v hodnote štatistiky markantné zlepšenie až o dva rády. Je potrebné upozorniť na rozdielnu silu testov pre rozdielne počty cifier pred a po odstránení anomálie. Túto skutočnosť budeme pozorovať aj na ďalších vizualizáciách.



Obr. 9: Porovnanie globálneho rozdelenia cifier v dátových rámcoch `L2` a `L2mod`.

Dataset	Použitý test	n_cifier	Hodnota štatistiky	df	p-value	PASS/FAIL
L2	Chi-squared	320000	$X^2 = 323452,71$	9	0,0000	FAIL
L2mod	Chi-squared	216911	$X^2 = 6651,29$	9	0,0000	FAIL

Tabuľka 3: Výsledky Testu 2.

- **Uniformita rozdelenia cifier a desatinných pozícií (Test 3 a 4)**

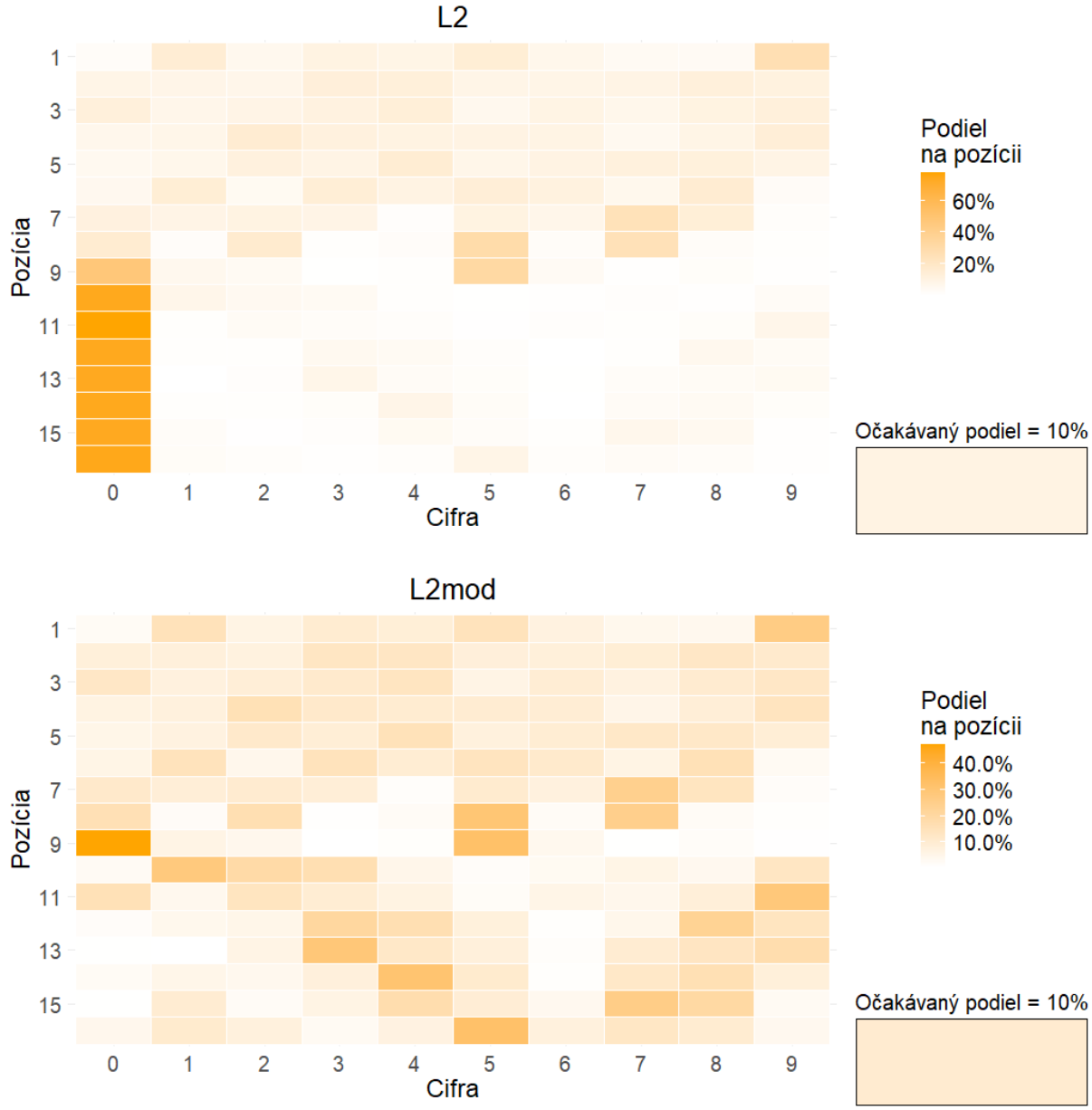
Ako pomôcku pre interpretáciu sme si vykreslili heatmapu, ktorú možno vidieť na Obrázku 10. Riadky v nej predstavujú konkrétne desatinné miesta a stĺpce cifry. Celkovo sa teda jedná o vizualizáciu zastúpenia každej cifry na každej pozícii. Čím tmavšia oranžová, tým má na danej pozícii daná cifra väčšie zastúpenie. Tu môžeme jasne vidieť anomáliu, ktorá nastala v datasete L2. Po jej odstránení dostávame lepší prehľad o povahe cifier aj po deviatej desatinnej pozícii.

V ideálnom prípade by sa mal každý obdĺžnik nachádzať v tesnej blízkosti hranice 10%, čo je jeho očakávaný podiel na každej pozícii. Po odstránení anomálie sa nám aj upravila mierka bližšie k očakávanej hodnote (i keď je stále priveľká na prejdenie testom). Pozrime sa na výsledky Testov 3 a 4, ktoré možno vidieť v Tabuľke 4.

Žiadna z cifier a pozícií neprešla ani jeden z testov, či už pred alebo po odstránení anomálie.

Dataset	Použitý test	n_cifier	n_PASS	which_PASS
L2	Chi-squared pos	320000	0/16	-
L2	Chi-squared dig	320000	0/10	-
L2mod	Chi-squared pos	216911	0/16	-
L2mod	Chi-squared dig	216911	0/10	-

Tabuľka 4: Výsledky Testov 3 a 4.

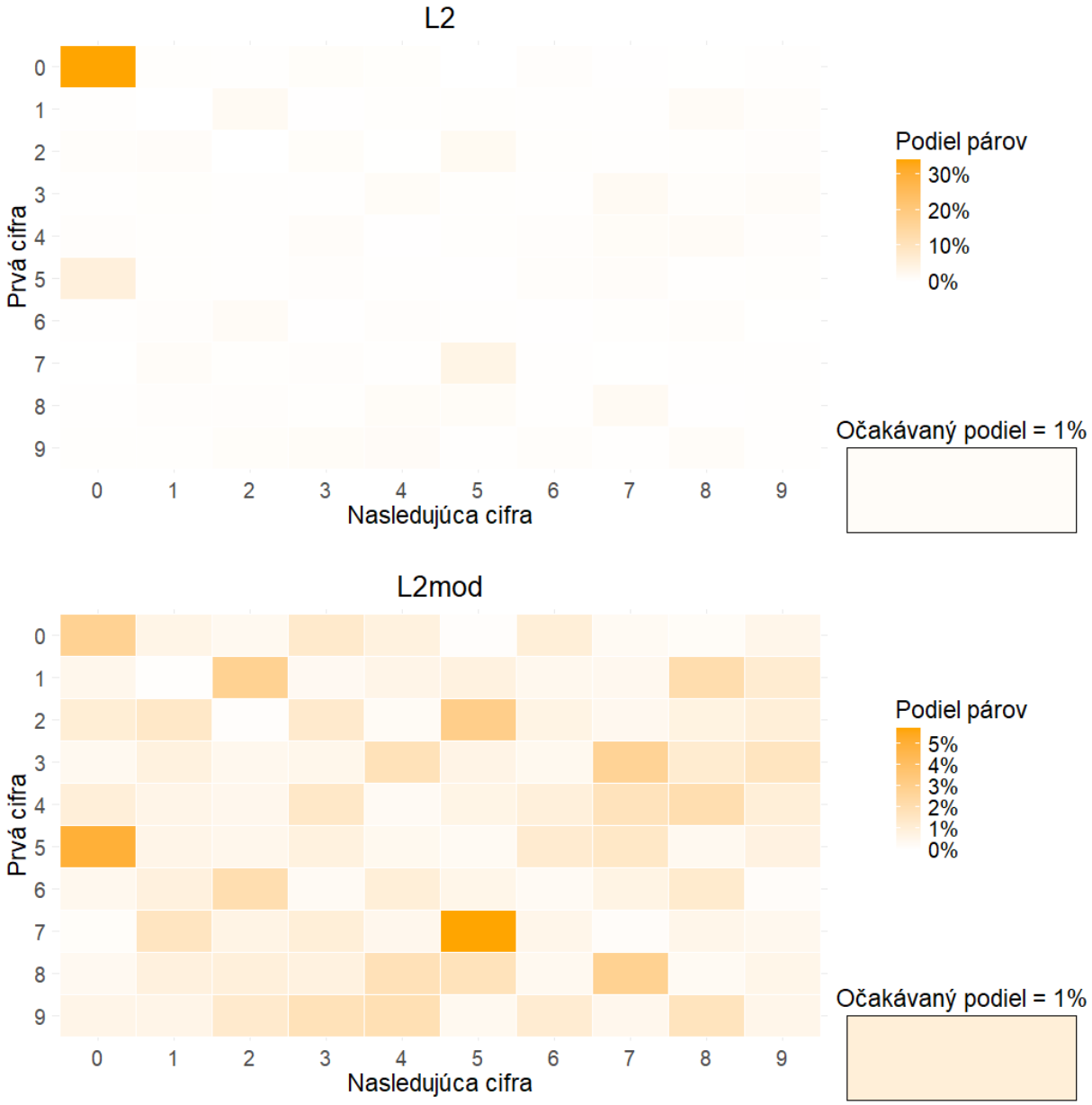


Obr. 10: Heatmapy cifier a pozícií pre `L2` a `L2mod`.

- **Náhodnosť dvojíc cifier (Test 5)**

Pri pohľade prvú heatmapu na Obrázku 11 datasetu `L2` toho veľa nevidíme. Až po odstránení anomálie môžeme skutočne sledovať povahu dát `L2`, ako vidíme na druhej heatmape. Vizualizácia sa číta nasledovným spôsobom: Obdlžnik na pozícii $(3, 2)$ predstavuje podiel páru číslic "21" zo všetkých párov vyskytujúcich sa v dátach `L2`/`L2mod`. Po odstránení anomálie zostávajú veľmi výrazné páry "50" a "75", pričom páry na diagonále sa javia ako menej zastúpené až na pár výnimiek. Diagonála je celkovo menej zastúpená než by sme očakávali.

Výsledky testu v Tabuľke 5 dopadli opäť ako sme čakali, všetky zlyhali. `L2mod` dosiahol o rád lepšiu hodnotu štatistiky, no stále je priveľká aj na 99 stupňov voľnosti.



Obr. 11: Heatmapy dvojíc cifier pre `L2` a `L2mod`.

Dataset	Použitý test	n_párov	Hodnota štatistiky	df	p-value	PASS/FAIL
`L2`	Chi-squared	300000	$X^2 = 3478792,96$	99	0,0000	FAIL
`L2mod`	Chi-squared	196911	$X^2 = 167925,60$	99	0,0000	FAIL

Tabuľka 5: Výsledky Testu 5.

- **Analýza a testovanie autokorelácií (Test 6)**

Začnime analýzou globálnej autokorelácie a parciálnej autokorelácie, tak ako ju spomíname v Teste 6). Na Obrázku 12 vidíme zobrazenie globálnych parciálnych autokorelácií až do lagu L = 17. Stĺpce zafarbené na červeno presahujú hranicu štatistickej signifikantnosti, čo indikuje, že dané lagy majú PACF štatisticky odlišnú od nuly pri zvolenej hladine významnosti $\alpha = 95\%$.

Hranice signifikantnosti pre PACF sa obyčajne odhadujú za predpokladu, že

$$\hat{\phi}_k \sim \mathcal{N}\big(0, \tfrac{1}{N}\big) \quad \text{pre veľké } N,$$

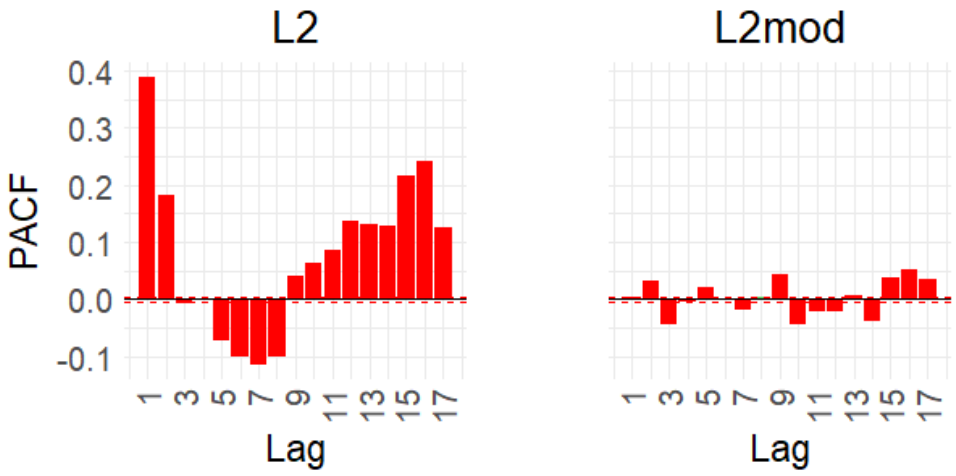
kde $\hat{\phi}_k$ predstavuje odhad parciálnej autokorelácie pre lag k [26]. Keďže N je v našom prípade aspoň 216 911, zvyčajne 320 000, tak pokladáme tento predpoklad za splnený pri platnosti H_0.

Pre zvolenú hladinu spoľahlivosti conf (u nás 0,95) sa určí kritická hodnota

$$\pm \frac{z_{1-(1-\text{conf})/2}}{\sqrt{N}}.$$

Hodnoty PACF, ktoré vybočia mimo tohto pásma považujeme za štatisticky významné. Stĺpce, ktoré majú hodnoty pod touto hranicou sú zafarbené na zeleno. Na analyzovanom Obrázku 12 teda vidíme veľký pokrok vo veľkosti PACF pre jednotlivé lagy. Tento pokrok možno vidieť aj na výsledku Ljung-Box testu v Tabuľke 6 keď sa pozrieme na hodnotu štatistiky. Opäť klesla o dva rády po oprave anomálie, čo značí výrazné priblíženie k očakávanej hodnote.

Pripravili sme si aj ďalšiu vizualizáciu, ktorá však nie je sprevádzaná žiadnym štatistickým testom pre jej povahu. Ako alternatívu k Obrázku 12 sme chceli ukázať aj parciálnu autokoreláciu v rámci čísel, nielen keď všetky desatinné cifry na seba nadviažeme do jedného časového radu.



Obr. 12: Histogramy globálnej PACF pre datasety `L2` a `L2mod`.

Dataset	Použitý test	n_cifier	Hodnota štatistiky	df	p-value	n_PASS	Lag
`L2`	Ljung-Box	320000	$Q = 331438,44$	17	0,0000	1/17	4
`L2mod`	Ljung-Box	216911	$Q = 3339,64$	17	0,0000	2/17	6, 8

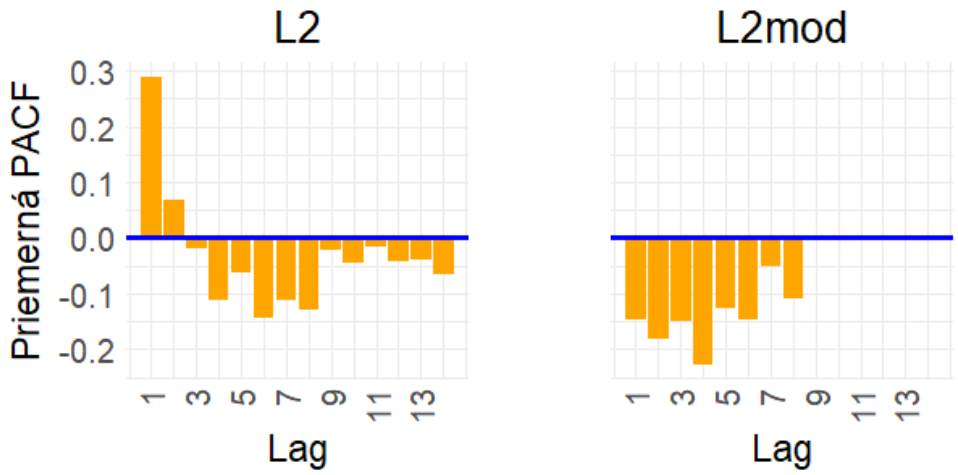
Tabuľka 6: Výsledky Testu 6.

Na Obrázku 13 porovnávame výsledky PACF spočítaných pre datasety `L2` a `L2mod`. Pre `L2` verziu sme vypočítali PACF až do lagu 14, najväčší možný lag pre 16 desatinných miest, ak chceme na výpočet každého lagu využiť aspoň dve hodnoty PACF. Pre verziu `L2mod` je PACF zobrazená iba do lagu 8, keďže by výpočet na číslach s desiatimi desatinnými miestami zlyhal (tie, kde bola odstránená anomália).

V prvom kroku prípravy tejto vizualizácie sme pre každé číslo z 20 000 zvlášť zavolali funkciu `pacf` a získali vektor hodnôt pre každý lag. Následne sme vypočítali priemer pacf naprieč všetkými číslami. Pre verziu `L2mod` sme potom prázdne hodnoty pre lagy 9 až 14 nahradili nulou, aby sa dala zachovať rovnaká os x pre jednoduchosť porovnávania.

Na výslednom dvojpanelovom grafe sme použili modrú horizontálnu čiaru v nule, ktorá opäť ukazuje ideálny prípad. Pri vytváraní tejto vizualizácie pre prípady neskôr ukázané v práci sme narazili na jednu nezrovnalosť, ktorú sme sa rozhodli prešetriť. Viaceré datasety vyzerali byť negatívne biasované s jasným vzorom, ktorý sa opakoval. Podľa [24] je empirický odhad pre autokorelácie aj parciálne autokorelácie jednoznačne vychýlený pre malú dĺžku časového radu. Viac sa preto analýze výsledkov tejto vizualizácie budeme

venovať neskôr v práci.



Obr. 13: Histogramy priemernej PACF v rámci čísel pre datasety `L2` a `L2mod`.

- **Otočné body v rámci čísel (Turning-Point Test) (Test 7)**

Test otočných bodov je jedným z dvoch komplexnejších experimentov v rámci explicitného testovania náhodnosti čísel v tejto práci. Na rozdiel od jednoduchých testov na uniformitu tu skúmame rozdelenie počtu tzv. otočných bodov T v každom čísle a porovnávame ho s presne vypočítaným teoretickým rozdelením (metodika popísaná v predchádzajúcej kapitole).

Na Obrázku 14 sú v oranžových stĺpcoch znázornené pozorované proporcie výskytu otočných bodov (očakávané frekvencie = pomery počtov O_t k celkovému počtu) pre datasety `L2` a `L2mod`. Modrá čiara ukazuje teoretické rozdelenie tejto štatistiky vypočítané pomocou dynamického programovania.

Pre `L2` je $P(T = t)$ priamo z distribúcie pre $n = 16$ cifier. Pre `L2mod` sme navyše zohľadnili frakciu

$$p = P\big(\text{posledných 7 cifier} = 0\big)$$

a vypočítali sme miešanú distribúciu

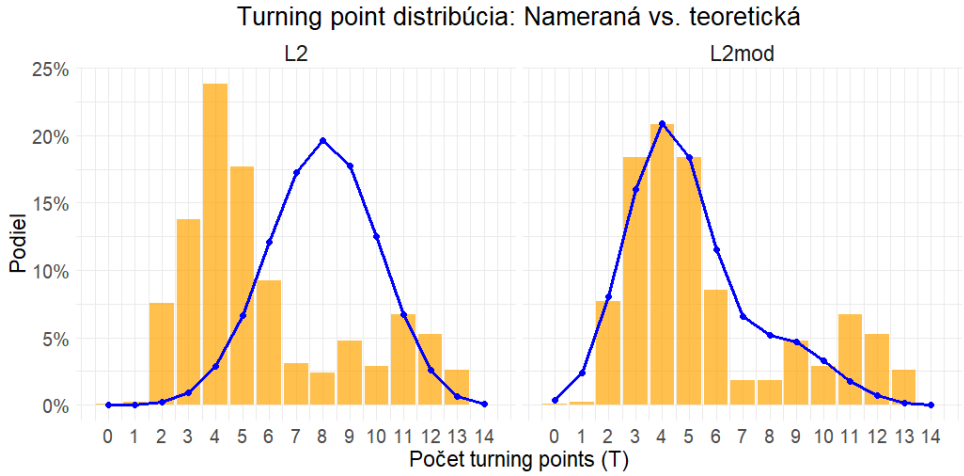
$$P_{\texttt{L2mod}}(T = t) = (1 - p)\, P_{16}(T = t) \;+\; p\, P_9(T = t),$$

kde P_9 je distribúcia pre skrátenú dĺžku $n = 9$ (po odstránení anomálie).

Ako môžeme vidieť, tak sa nezmenili iba hodnoty teoretických pravdepodobností, ale aj pozorované hodnoty O_t. Toto nastalo z dôvodu skrátenia veľkého počtu jednotlivých čísel na dĺžku deviatich desatinných miest.

Dataset `L2` je naozaj veľmi vzdialený od teoretickej distribúcie pre 16 desatinných miest. Počet otočných bodov je umelo znížený výskytom anomálie. Reálne rozdelenie datasetu môžeme teda vidieť na pravej časti tejto vizualizácie, kde sa nám podarilo účinne anomáliu odstrániť.

V `L2mod` stále však vidíme primálo otočných bodov na hodnotách 7 a 8. Privysoké hodnoty tejto štatistiky sú zase pri počtoch 11, 12 a 13. Budeme tieto hodnoty sledovať aj pri iných datasetoch.



Obr. 14: Pozorované distribúcie počtu otočných bodov pre datasety `L2` a `L2mod`.

V Tabuľke 7 sú vyobrazené výsledky Testu 7. Stĺpec "Pomer" ukazuje aká časť čísel bola postihnutá anomáliou. Toto číslo je väčšie než sme prvotne predpokladali a vysvetľuje dramatický posun očakávaného rozdelenia na Obrázku 14. Hodnota štatistiky sa taktiež zlepšila o celý rád, no nestačilo to na prejdenie testom.

Dataset	Použitý test	Hodnota štatistiky	df	Pomer	p-value	PASS/FAIL
`L2`	Chi-squared	$X^2 = 132780,10$	13	-	0,0000	FAIL
`L2mod`	Chi-squared	$X^2 = 17110,07$	13	0,736	0,0000	FAIL

Tabuľka 7: Výsledky Testu 7.

- **Rozdeľovanie čísel do tried**

Experiment rozdeľovania čísel do tried je druhý komplexnejší experiment, ktorý sme vymysleli. Motiváciou pre jeho tvorbu bola istá podobnosť čísel v rámci datasetov `L1` až `L3`, ktorú sme si všimli počas generovania rámcov v GPT4. Generované čísla neboli zväčša úplne identické, no mali medzi sebou veľa podobností.

Náš algoritmus iteratívne spracúva každé číslo a rozhoduje o jeho začlenení takto:

1. **Porovnanie s existujúcimi triedami:** Pre každú už vytvorenú triedu sa vyberie "reprezentant" (v našom prípade prvé číslo z danej triedy). Tento reprezentant sa pre všetky jeho cyklické rotácie (napríklad $0,12345 \rightarrow 0,23451;\ 0,34512;\ldots$) porovná s aktuálnym číslom. Následne spočítame koľko desatinných pozícií sa zhoduje s aktuálnym číslom v každej z rotácií, a z toho vyberieme maximálny počet zhôd v rámci triedy a následne aj najväčší počet zhôd spomedzi všetkých testovaných tried.

2. **Priradenie alebo založenie novej triedy:** Ak maximálny počet zhôd presiahne daný prah `threshold` (u nás 6), číslo sa priradí do triedy, ktorej reprezentant s ním dosiahol túto maximálnu zhodu. V opačnom prípade sa pre číslo vytvorí nová samostatná trieda.

3. **Aktualizácia počtov zhôd:** Každé priradenie zvyšuje príslušnú diagonálnu hodnotu v matici `similarity`, čiže kardinalitu tejto triedy v našom zápise. Zároveň pre každú inú triedu, ktorej reprezentant nameral počet zhôd nad prahom, sa symetricky zvýši príslušný mimo-diagonálny prvok (spoločné zhody medzi triedami) o 1.

4. **Rotácie a práca s reprezentantom:** Rotácie zabezpečujú, že dve čísla, ktoré si budú "podobné" už len posunutím dekadického reťazca, budú stále zaradené do rovnakej triedy. Použitie len jedného reprezentanta na triedu udržuje porovnávanie rýchle za cenu exaktnosti. Tento výber bol spravený pre výpočtové dôvody.

5. **Výsledok:** Na konci dostaneme množinu tried, v ktorej každá obsahuje čísla, ktoré sú navzájom dostatočne podobné podľa definovaného prahu. Ďalej vieme vyčísliť veľkosti tried a aj podobnosť tried medzi sebou vďaka matici `similarity`.

Ako príklad uvádzame sekvenciu čísel v Tabuľke 8a. Táto sekvencia je priamo vytiahnutá z rámca `L3` a maticu podobnosti s kardinalitami na diagonále v Tabuľke 8b. Ako prah sme v našom testovaní zvolili prah = 6, aby sme vedeli medzi sebou porovnávať aj `L2mod` s inými datasetmi. Ak by sme totiž zvolili ako prah číslo 10 a viac, tak by sme nemohli ani zmerať rámec `L2mod` v tejto metrike.

Index	Číslo	Trieda
8000	0,7525938464939265	A
8001	0,4729163859271834	B
8002	0,7381937264829372	A
8003	0,3174827364839162	A
8004	0,6612848374929363	C
8005	0,4917163748371628	B
8006	0,8215827165829364	C
8007	0,3839275639372643	A
8008	0,8134957261839264	A
8009	0,4295828371937262	A

(a) Príklad klasifikácie čísel do tried.

Triedy	A	B	C
A	6	1	1
B	1	2	0
C	1	0	2

(b) Matica podobnosti pre triedy z príkladu.

Tabuľka 8: Ukážka metodológie zaraďovania do tried na príklade.

Pozrime sa na nami vytvorené vizualizácie pre túto metriku. Pri vizualizáciách sme pracovali výlučne s nami vytvorenou maticou podobnosti, ktorej príklad môžeme vidieť v Tabuľke 8b.

Ako prvú vizualizáciu sme pripravili metrické MDS (Multi-dimensional scaling). Metrické MDS je technika, ktorá umožňuje zobraziť objekty so známymi vzájomnými hodnotami vzdialenosti v nízkorozmernom Euklidovskom priestore. Cieľom je nájsť súradnice bodov $x_1, \ldots, x_n \in \mathbb{R}^2$ tak, aby rozdiely medzi zadanými hodnotami vzdialenosti d_{ij} a Euklidovskými vzdialenosťami $\|x_i - x_j\|$ boli čo najmenšie. Minimalizuje sa tzv. stress funkcia

$$\sum_{i<j} \big(d_{ij} - \|x_i - x_j\|\big)^2,$$

vďaka čomu dostaneme mapovanie, ktoré čo najvernejšie zachováva vzájomné vzdialenosti

medzi objektmi.

V našom experimente sme však nezačínali so vzdialenosťami, ale s maticou podobností $\mathrm{sim}(i, j)$, ktorá vyjadruje, koľko zhôd sa vyskytlo medzi dvoma triedami. Aby sme túto maticu mohli použiť v MDS, museli sme ju pretransformovať na maticu vzdialenosti. Zvolili sme jednoduchú transformáciu

$$d(i, j) \;=\; \frac{1}{\mathrm{sim}(i, j) + 1} \quad \text{s bodmi } d(i, i) = 0.$$

Takáto transformácia pre pár bez zhôd ($\mathrm{sim} = 0$) vráti $d = 1$, čo je maximálna možná vzdialenosť, a so stúpajúcou podobnosťou postupne klesá k nule.

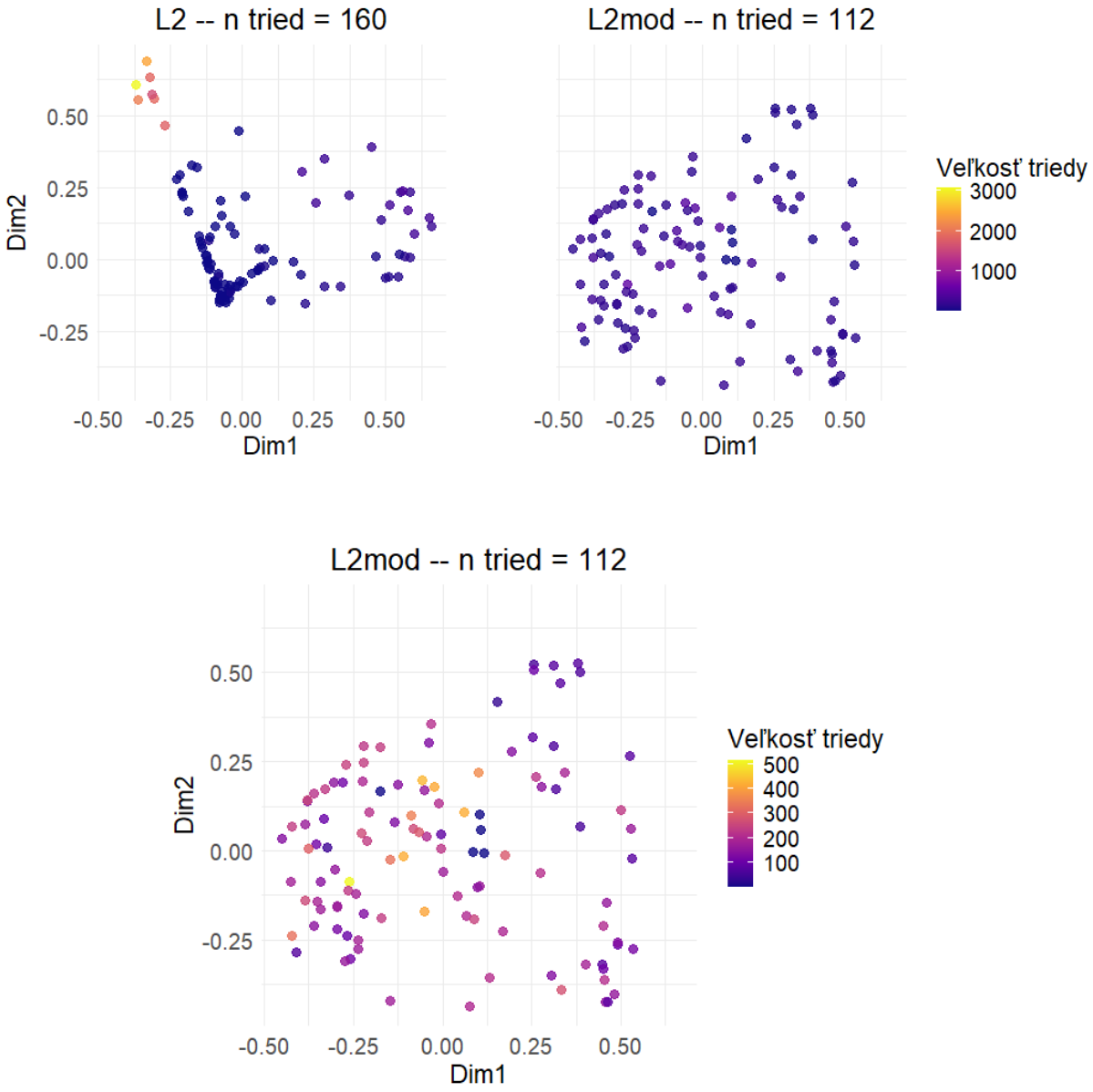
Prečo sme sa rozhodli práve pre $d = 1/(\mathrm{sim}+1)$? Predovšetkým preto, lebo táto forma zaručuje, že všetky hodnoty vzdialenosti ležia v jednotkovom intervale $[0; 1]$, bez ohľadu na absolútnu veľkosť pôvodných podobností. Zachováva monotónnosť poradia párov podľa podobnosti a zároveň je výpočtovo jednoduchá a numericky stabilná. Vďaka tomu vieme priamo porovnať maticu podobností z rôznych datasetov, aj keby mali veľmi rôzne škály počtov zhôd a získať konzistentné vstupy pre MDS.

Na Obrázku 15 môžeme vidieť toto zobrazenie pre `L2` a `L2mod`. Vidíme, že `L2` má zhluk siedmych veľkých (kardinalitou) tried, ktoré si sú aj veľmi podobné. Po analýze v dátach aj vyšlo, že práve týchto sedem tried obsahuje anomáliu. Výskyt týchto siedmych tried nám následne aj veľmi skresľuje celé zobrazenie v 2D. Po odstránení anomálie získavame zobrazenie `L2mod`, ktoré je následne aj zobrazené vo vlastnej mierke na obrázku nižšie. Žiadne ďalšie výrazné zhluky sa tu už nenachádzajú.

Ďalšia vizualizácia, ktorú sme vytvorili pre túto metriku je dendrogram, ktorý vznikol po hierarchickom zhlukovaní matice vzdialenosti. Táto matica vzdialenosti vznikla rovnakým spôsobom ako v MDS.

Hierarchické zhlukovanie prirodzene vytvára stromovú štruktúru (dendrogram), v ktorej každý list predstavuje jednu triedu z pôvodnej matice podobností a vnútorné uzly zlučujú dve podtriedy na základe ich vzájomnej vzdialenosti. Hierarchické zhlukovanie metódou priemerných vzdialeností “average linkage” potom postupne spája tie dve triedy alebo zhluky tried, ktorých priemerná vzdialenosť je najmenšia.

Na výslednom dendrograme výška každého spoja zodpovedá vzdialenosti, pri ktorej sa dva zhluky/dve triedy zlúčili. Čím nižšie (bližšie k nule) je zlučovací uzol, tým vyššiu podobnosť medzi týmito dvomi skupinami sme zaznamenali.



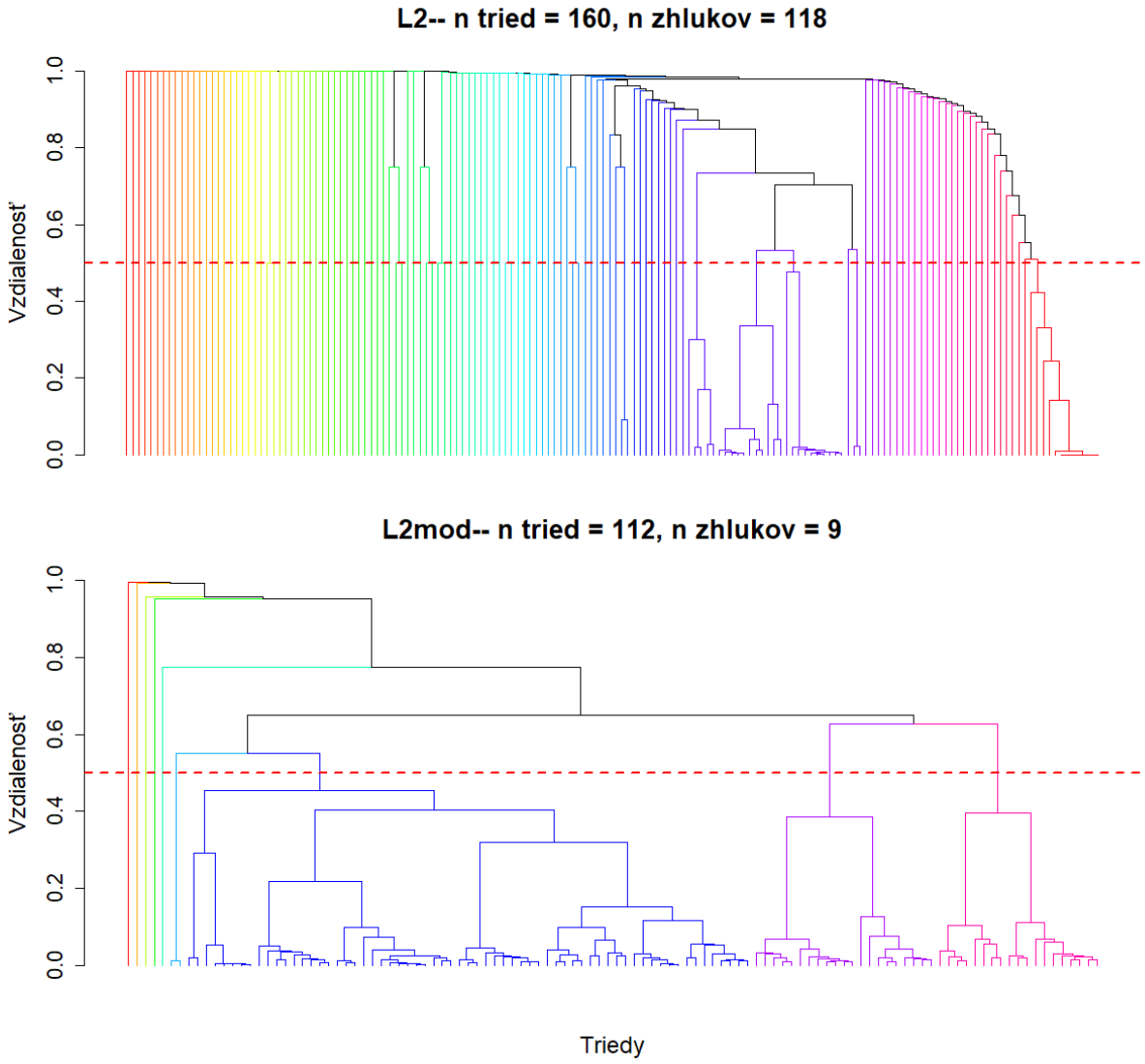
Obr. 15: MDS vizualizácia pre `L2` a `L2mod` v rôznych mierkach.

Naopak, spoje blízko hodnoty 1 označujú zhluky, ktoré do tejto chvíle neprejavili takmer žiadnu zhodu.

Aby sme mohli vizuálne oddeliť zhluky s dostatočne vysokou vnútornou podobnosťou, nakreslili sme na každom dendrograme červenú prerušovanú čiaru v hladine `cut_height` = 0, 5. Táto prahová úroveň nám umožňuje interpretovať všetky vetvy pod čiarou ako opodstatnené zhluky — zhlukovanie nad ňou ponecháva triedy, ktoré sú medzi sebou príliš odlišné na to, aby sme ich považovali za jednu skupinu. Výsledkom sú grafy, kde každá fa-

rebne odlíšená vetva reprezentuje jednu homogénnu skupinu tried so silnejšími vnútornými podobnosťami a slabšími väzbami na ostatné zhluky.

Prvú ukážku dendrogramu môžeme vidieť na Obrázku 16, kde porovnávame `L2` a `L2mod`. V `L2mod` vidíme viac zo skutočnej povahy dát, keďže sú už očistené o anomáliu.



Obr. 16: Dendrogramy rozdelenia tried pre `L2` a `L2mod`.

3.1.3 Porovnanie všetkých finálnych datasetov

Štruktúra každej série vizualizácií v tejto podkapitole bude nasledovať túto štruktúru: Najskôr bude zobrazený panel s datasetmi `L1`, `L2mod`, `L3` a `MT`, kde budeme vedieť dobre porovnať tieto štyri datasety pri rovnakých mierkach. Následne budeme pokračovať datasetmi `MT` a `WH`, ktoré sú veľmi podobné vo svojej povahe. Rozdiely medzi týmito dvoma sadami by neboli viditeľné na rovnakej mierke zvolenej pre prvé štyri datasety, preto sme sa rozhodli ich oddeliť do svojej vlastnej vizualizácie.

Podstata vizualizácií už bola vysvetlená bližšie v Podkapitole 3.1.2, preto sa budeme venovať primárne interpretácii výsledkov.

• **Overenie rovnomerného rozdelenia hodnôt na** $[0; 1]$ **(Test 1)**

Na Obrázku 17 môžeme vidieť rozloženie čísel v dátových sadách `L1` až `MT`, pre počty binov nbins $= 10, 30, 100$.

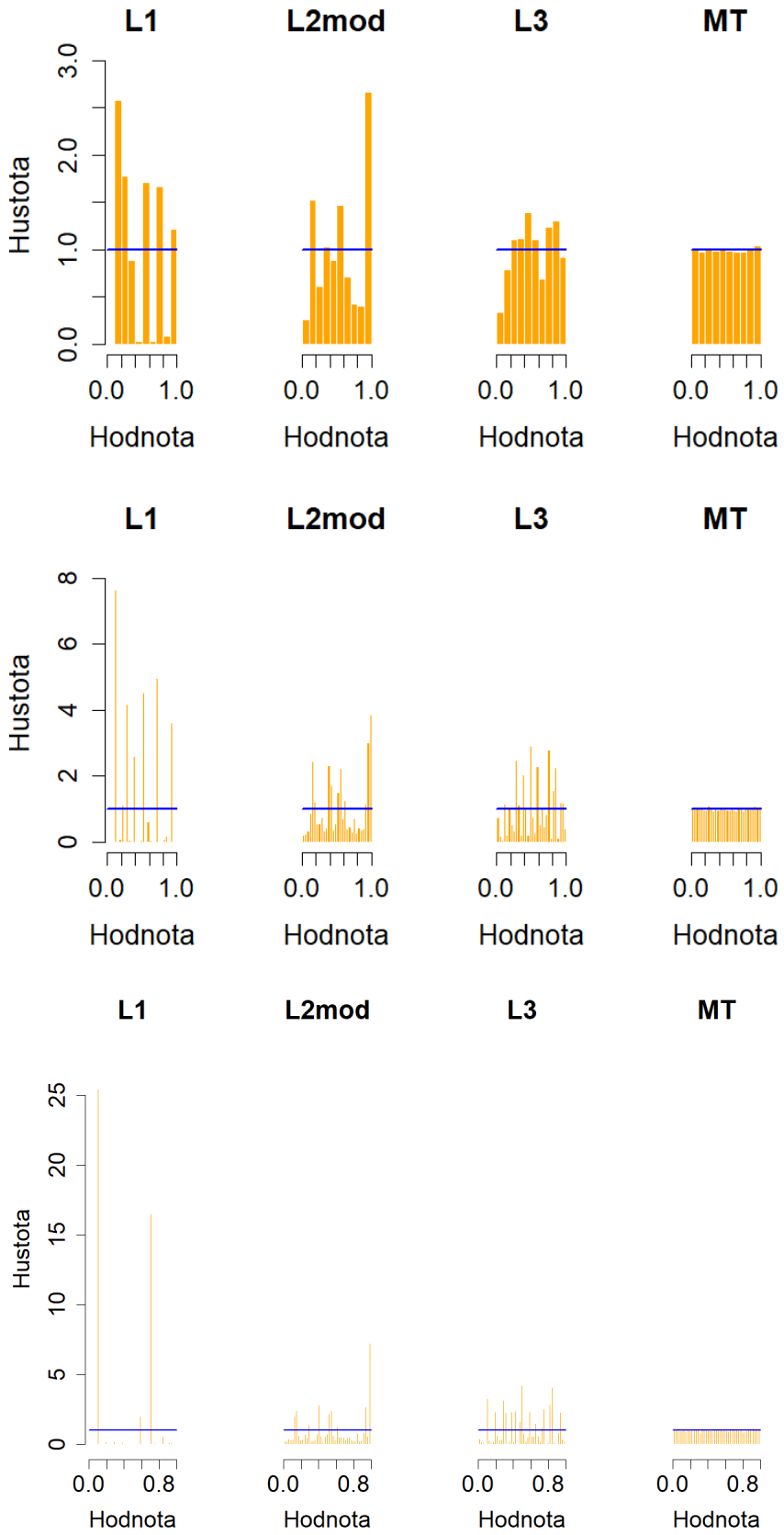
Dataset `L1` sa ani len nepriblížil rovnomernosti v tejto vizualizácii, so zvyšujúcim počtom bins sa aj dramaticky zvyšovalo zastúpenie hodnôt v už aj tak vysoko zastúpených kategóriách. Upozornili by sme na absenciu čísel, ktoré spadli do kategórií pod $0, 1$, ako aj dalšie kategórie.

Dataset `L3` vyzerá voľným okom najlepšie spomedzi rámcov generovaných GPT4, tiež je však na ňom vidno malé zastúpenie hodnôt pod $0, 1$. Rovnakú vlastnosť má aj dataset `L2mod`. `MT` dodržuje očakávanú hodnotu pri tejto mierke takmer dokonalo.

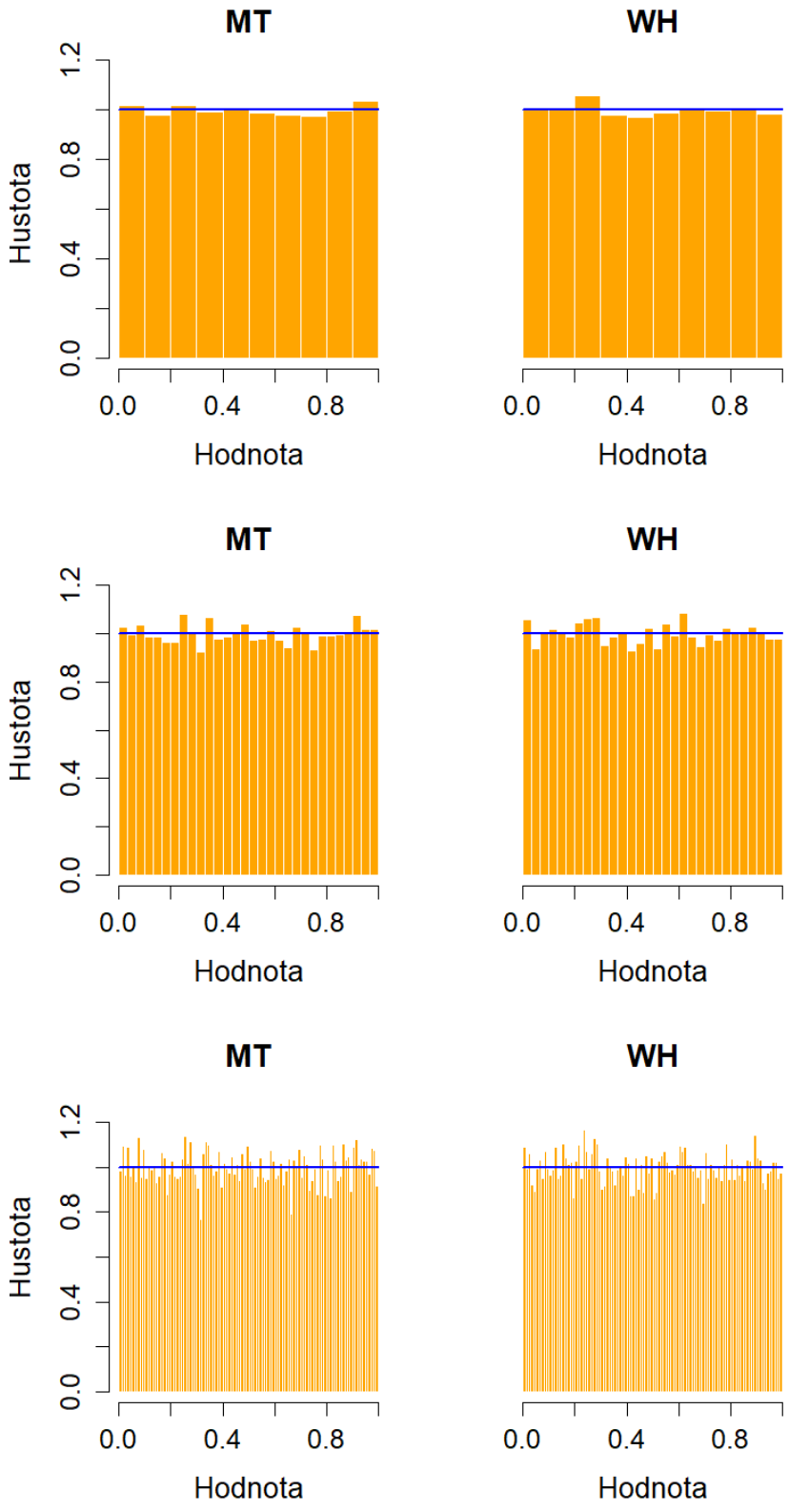
Na Obrázku 18 môžeme vidieť rozloženie čísel v dátových sadách `MT` a `WH`, pre počty binov nbins $= 10, 30, 100$.

Okamžite si môžeme všimnúť dramatickú zmenu v mierke. Táto mierka sa dokonca ani nemení pre rôzne počty binov, čo ukazuje na vysokú robustnosť datasetov `MT` a `WH`. Ku zásadným výkyvom nikde nedochádza.

Výsledky testov možno vidieť v Tabuľke 9. Testy sme pre prehľadnosť vykonávali už iba pre 30 binov. Náš odhad pre hodnoty štatistík sedel pre rámce generované GPT4 – `L3` malo najlepší výsledok. Žiadne z nich však neprešli pre 29 stupňov voľnosti. Oba `MT` aj `WH` prešli testami. Pre `MT` a `WH` bol aktivovaný scenár použitia KS testu, keďže sa v nich nenachádzali žiadne zhody.



Obr. 17: Rozmiestnenie hodnôt `L1` – `MT` na intervale $[0; 1]$ pre rôzne počty binov.



Obr. 18: Rozmiestnenie hodnôt `MT` a `WH` na intervale $[0; 1]$ pre rôzne počty binov.

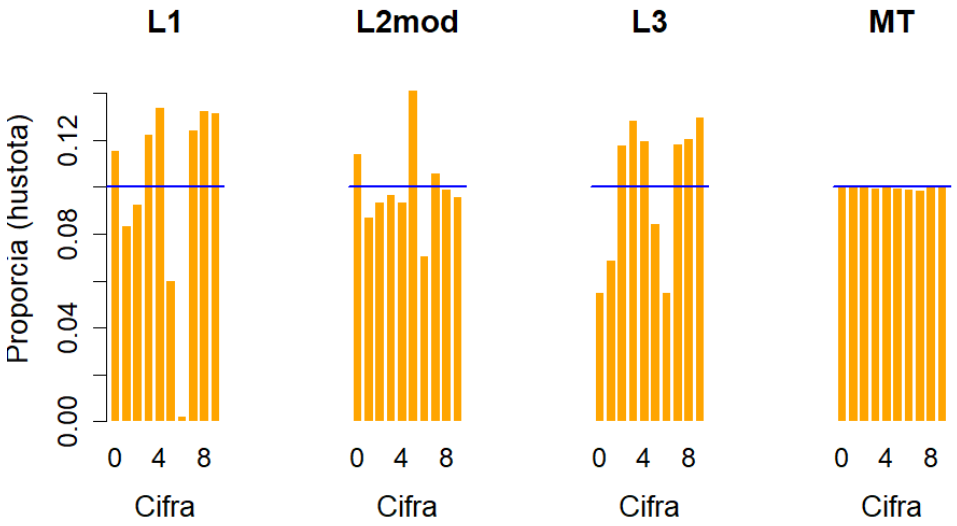
Dataset	Použitý test	n_bins	Hodnota štatistiky	df	p-value	PASS/FAIL
L1	Chi-squared	30	$X^2 = 75325,91$	29	0,0000	FAIL
L2mod	Chi-squared	30	$X^2 = 16574,64$	29	0,0000	FAIL
L3	Chi-squared	30	$X^2 = 13855,25$	29	0,0000	FAIL
MT	KS	-	$D = 0,0051$	-	0,6685	PASS
WH	KS	-	$D = 0,0069$	-	0,2983	PASS

Tabuľka 9: Výsledky Testu 1 pre n = 20000.

- **Overenie globálneho rovnomerného rozdelenia cifier (Test 2)**

Na Obrázku 19 môžeme vidieť rozloženie čísel v dátových sadách L1 až MT.

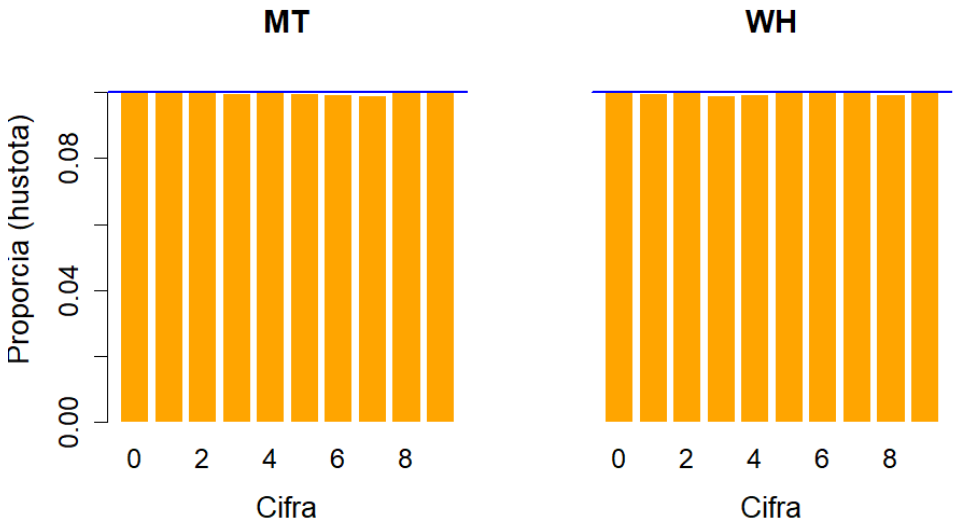
V tejto vizualizácii už vyzerajú byť datasety L1 až L3 omnoho podobnejšie než predtým. Veľmi zaujímavá je absencia číslice 6, ktorá má dramaticky nižšie zastúpenie vo všetkých GPT4 datasetoch. MT sa správa ako by sme čakali.



Obr. 19: Porovnanie globálneho rozdelenia cifier v dátových rámcoch L1 – MT.

Priblížme si teraz datasety MT a WH na Obrázku 20. Vyzerajú bezchybne, veľmi rovnomerne rozložené cifry.

V Tabuľke 10 môžeme vidieť výsledky Testu 2. L2mod vyhral s väčším vedením než sme čakali spomedzi nami generovaných datasetov, no žiaden z nich neprešiel testom. MT a WH oba prešli bez problémov.



Obr. 20: Porovnanie globálneho rozdelenia cifier v dátových rámcoch `MT` a `WH`.

Dataset	Použitý test	n_cifier	Hodnota štatistiky	df	p-value	PASS/FAIL
`L1`	Chi-squared	320000	$X^2 = 51442,73$	9	0,0000	FAIL
`L2mod`	Chi-squared	216911	$X^2 = 6651,29$	9	0,0000	FAIL
`L3`	Chi-squared	320000	$X^2 = 27024,86$	9	0,0000	FAIL
`MT`	Chi-squared	320000	$X^2 = 8,08$	9	0,5256	PASS
`WH`	Chi-squared	320000	$X^2 = 7,36$	9	0,5992	PASS

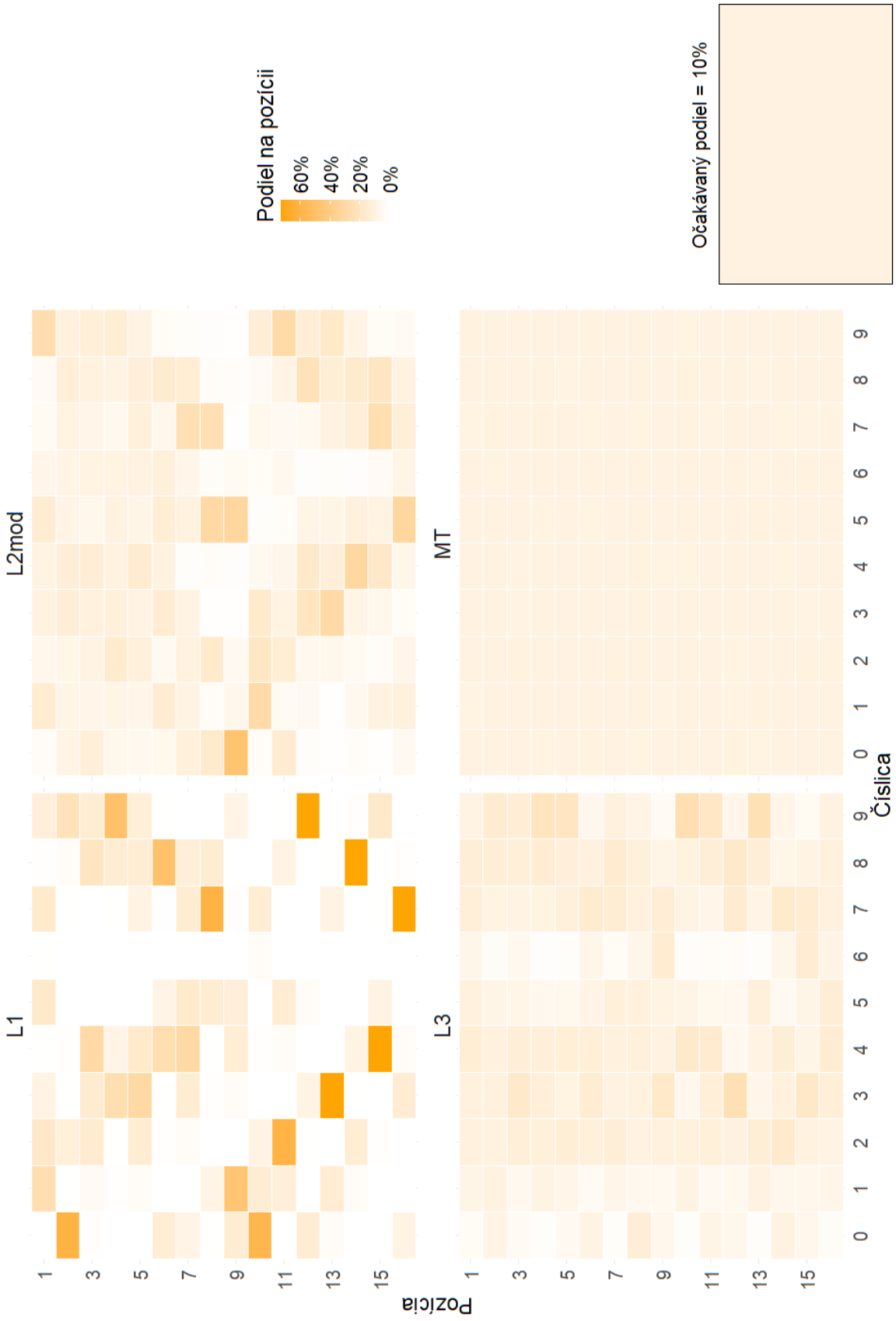
Tabuľka 10: Výsledky Testu 2.

- **Uniformita rozdelenia cifier a desatinných pozícií (Test 3 a 4)**

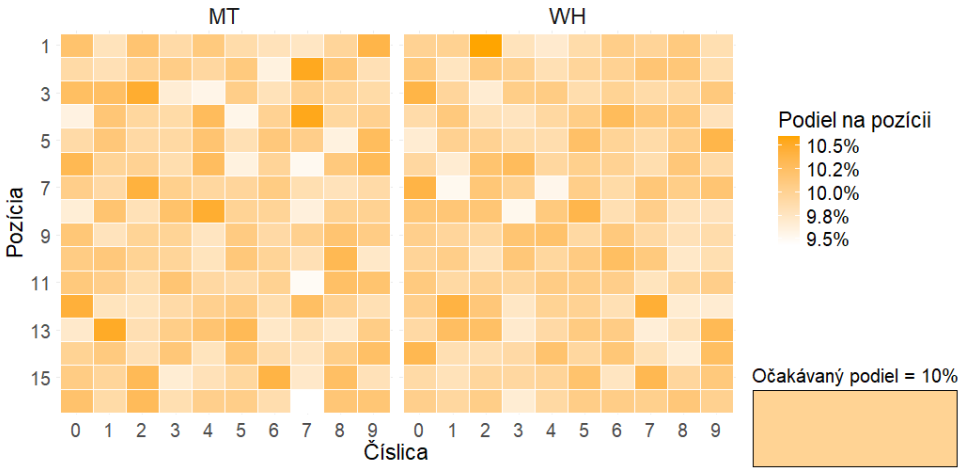
Na Obrázku 21 môžeme vidieť rozloženie cifier na pozíciách v dátových sadách `L1` až `MT`.

Dataset `L1` má niektoré pozície takmer exkluzívne obsadené jedinou cifrou. Medzi po sebe idúcimi pozíciami je vidno akúsi osciláciu medzi nízkymi ciframi a vysokými. Teraz je krásne vidno absenciu cifry 6 pre všetky datasety generované GPT4. Pozostatky podobného správania ako má `L1` sú vidno aj v `L2mod` v druhej polovici sledovaných cifier (osciláciu). Dataset `L3` vyzerá určite najlepšie spomedzi `L1`-`L3`. Jednoznačný problém v ňom vidíme len pri cifrách 0 a 6.

Pozrime sa bližšie na dataset `MT` v porovnaní s `WH` na Obrázku 22. Už zo zmeny mierky vidíme, že sa pohybujeme na lepších hodnotách ako doteraz. Problémová môže byť cifra 7 v datasete `MT` alebo pozícia 7 v `WH`, no nevieme to určiť definitívne.



Obr. 21: Heatmapy cifier a pozícií pre `L1` – `MT`.



Obr. 22: Heatmapy cifier a pozícií pre `MT` a `WH`.

V Tabuľke 11 vidíme výsledky oboch testov, aj s konkrétnymi pomenovaniami pozícií/cifier, ktoré prešli testami. Opäť ani jeden z nami generovaných datasetov neprešiel, no tentokrát nastal malý problém aj v datasete `MT` pri jednej z cifier, konkrétne cifra 7, ako sme predvídali.

Ako teda vyhodnotiť tento test pre `MT`? Máme viacero možností:

1. **Striktný prístup:** Aj jediná nesprávna cifra znamená zamietnutie nulovej hypotézy, takže aj `MT` by bol klasifikovaný ako *FAIL*.

2. **Korekcia na viacnásobné testovanie:** Aplikovaním Bonferroniho korekcie [25] ($\alpha' = \alpha/10 = 0{,}005$) sa prísnosť zvyšuje – potom by ani jedna z desiatich testovaných číslic nesmela padnúť. V takom prípade by `MT` zlyhal ešte väčšmi.

3. **Tolerantný prístup:** Pre "dig" test sme očakávali desať nezávislých testov (jeden pre každú cifru). Pri hladine $\alpha = 0{,}05$ je očakávaný počet chýb $10 \cdot 0{,}05 = 0{,}5$. Jedna nesprávna cifra "7" teda nevybočuje významne z tohto očakávania, takže by sme `MT` mohli považovať za *marginal pass*.

Vzhľadom na to, že `MT` zlyhal len pri jednej z desiatich číslic a počet chýb je približne v súlade s očakávaným počtom chýb 0,5 pre $\alpha = 0{,}05$, odporúčame pri interpretácii klasifikovať `MT` ako "*marginal pass*" a označiť výsledok žltou farbou v tabuľke.

Definujeme teda tri stupne výsledku:

- *PASS* – pre datasety s 0 chybami,

- *MARGINAL* – pre datasety s 1–2 chybami (ako `MT`),
- *FAIL* – pre viac ako 2 chyby.

Dataset	Použitý test	n_cifier	n_PASS	which_PASS
`L1`	Chi-squared pos	320000	0/16	-
`L1`	Chi-squared dig	320000	0/10	-
`L2mod`	Chi-squared pos	216911	0/16	-
`L2mod`	Chi-squared dig	216911	0/10	-
`L3`	Chi-squared pos	320000	0/16	-
`L3`	Chi-squared dig	320000	0/10	-
`MT`	Chi-squared pos	320000	16/16	ALL
`MT`	Chi-squared dig	320000	9/10	0,1,2,3,4,5,6,8,9
`WH`	Chi-squared pos	320000	16/16	ALL
`WH`	Chi-squared dig	320000	10/10	ALL

Tabuľka 11: Výsledky Testu 3 a 4.

- **Náhodnosť dvojíc cifier (Test 5)**

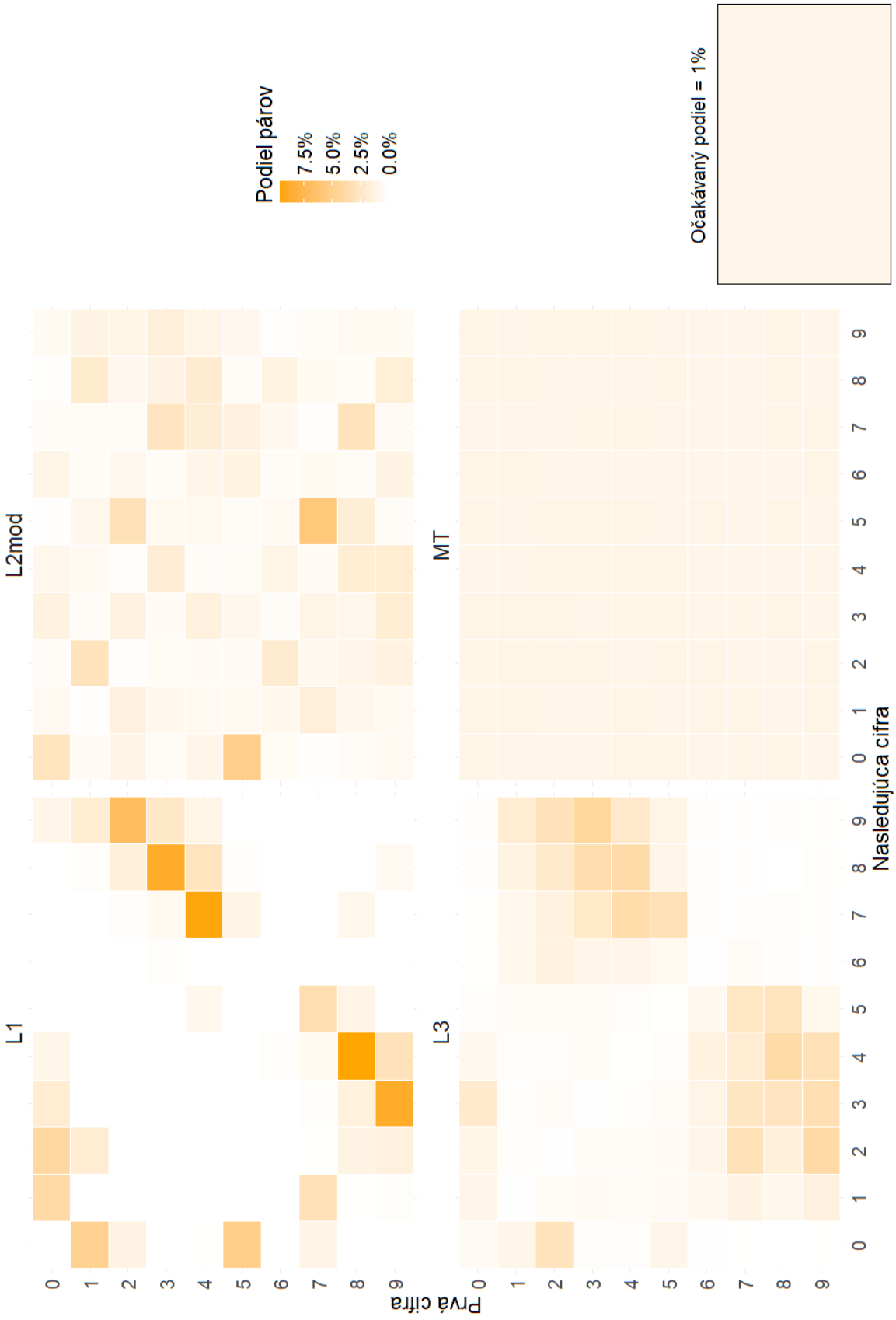
Na Obrázku 23 môžeme vidieť rozloženie párov cifier v dátových sadách `L1` až `MT`.

Následnosť cifier nám je hneď podozrivá v datasetoch `L1` a `L3`. `L3` vyzerá ako miernejšia verzia `L1`, no stále preukazuje podobné zastúpenie dvojíc. Pri `L1` sme museli ošetriť aj neočakávaný patologický prípad – niektoré páry nenastali ani raz v celom datasete.

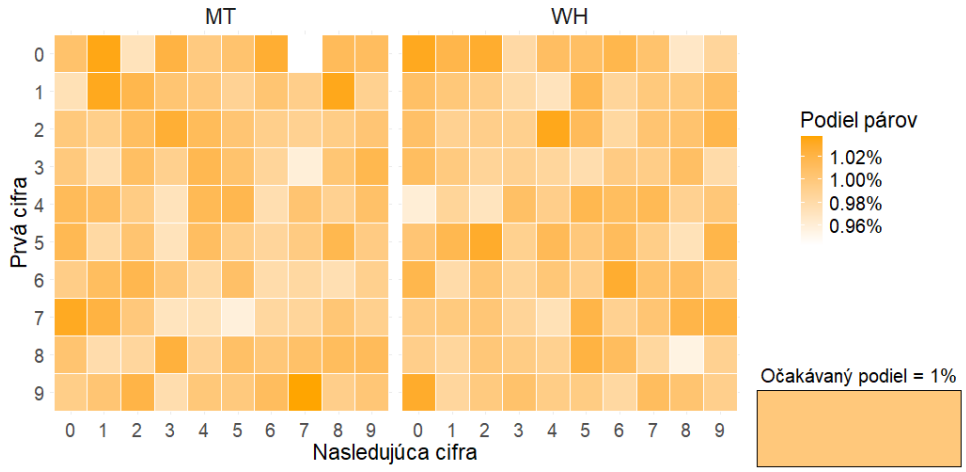
`L2mod` sa akosi vyhlo spomínanému vzoru. Jasne vieme ukázať iba na pár "75", ktorého početnosť vyzerá byť dramaticky prehnaná. Diagonála je málo zastúpená vo všetkých datasetoch generovaných LLM.

Pozrime sa bližšie na datasety `MT` a `WH` na Obrázku 24. Nevieme s určitosťou poukázať na žiaden pár s veľkou istotou, je potrebné sa pozrieť na výsledky testov v Tabuľke 12. Aj `MT` aj `WH` s prehľadom prešli týmto testom.

Z datasetov `L1`-`L3` obstál najlepšie `L2mod`, no stále je veľmi ďaleko od počtu df. Tento test je naozaj veľmi náročný na splnenie pre LLM v našom testovaní.



Obr. 23: Heatmapy dvojíc cifier pre `L1` – `MT`.



Obr. 24: Heatmapy dvojíc cifier pre `MT` a `WH`.

Dataset	Použitý test	n_párov	Hodnota štatistiky	df	p-value	PASS/FAIL
`L1`	Chi-squared	300000	$X^2 = 1307487,73$	99	0,0000	FAIL
`L2mod`	Chi-squared	196911	$X^2 = 167925,60$	99	0,0000	FAIL
`L3`	Chi-squared	300000	$X^2 = 385220,37$	99	0,0000	FAIL
`MT`	Chi-squared	300000	$X^2 = 92,63$	99	0,6608	PASS
`WH`	Chi-squared	300000	$X^2 = 75,89$	99	0,9592	PASS

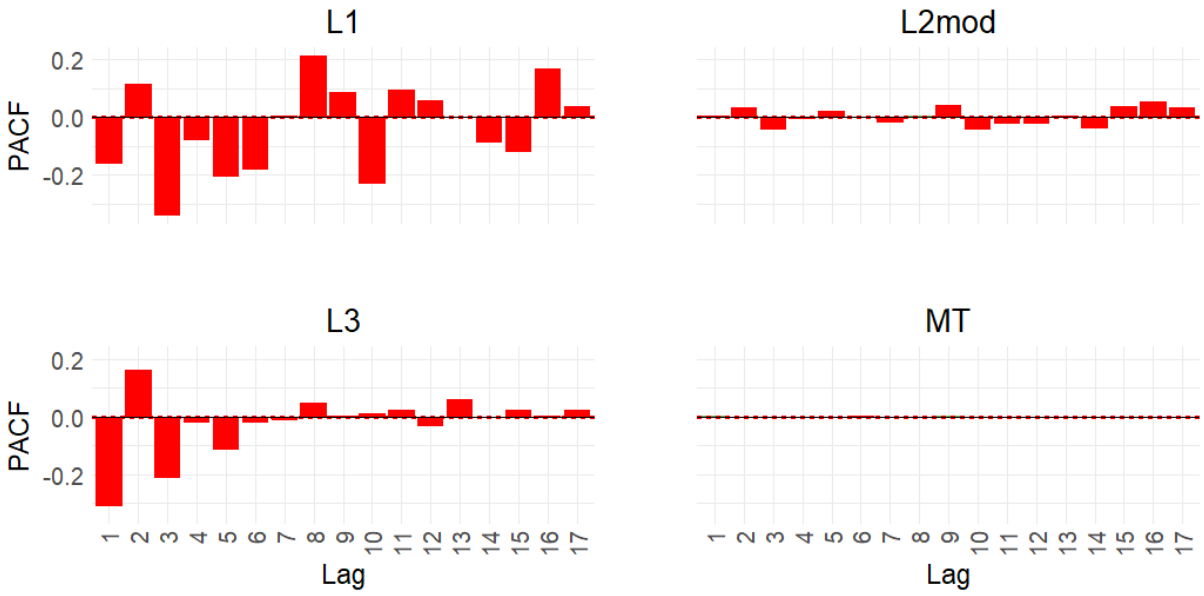
Tabuľka 12: Výsledky Testu 5.

- **Analýza a testovanie autokorelácii (Test 6)**

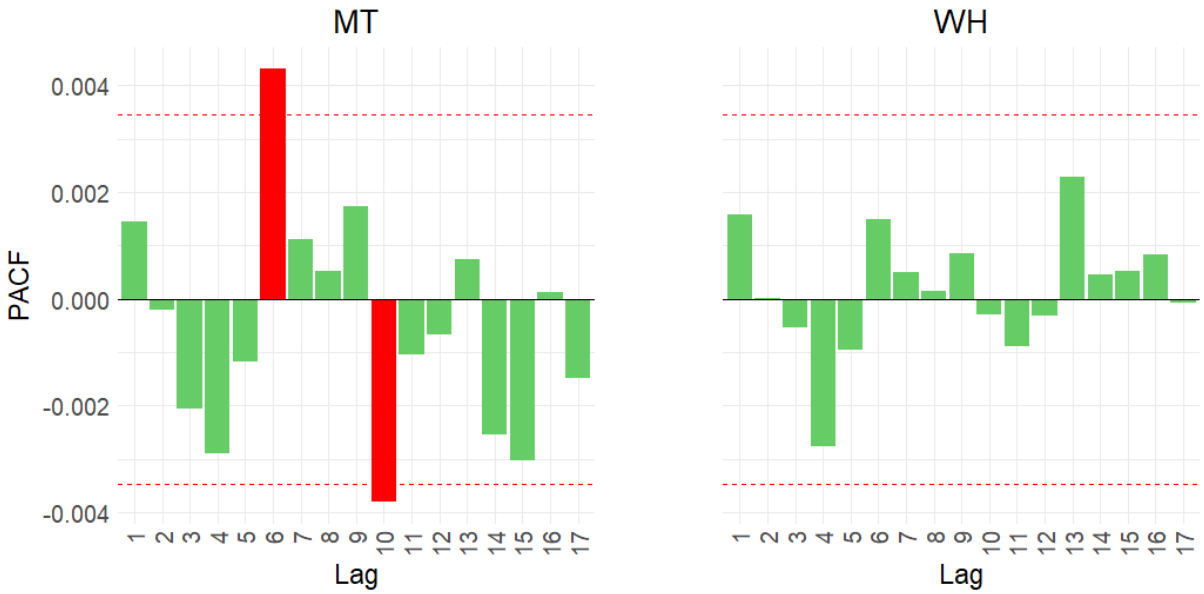
Na Obrázku 25 môžeme vidieť globálne parciálne autokorelácie pre `L1` až `MT`. `L1` a `L3` majú veľmi výrazné PACF pre viacero lagov. Hodnoty PACF pre `L3` sa postupne znižujú pre rastúci lag. `MT` majú zanedbateľnú hodnotu PACF, uvidíme lepšie na bližšom porovnaní.

Pozrime sa na bližšie porovnanie `MT` a `WH` pre globálne parciálne autokorelácie na Obrázku 26. Prvýkrát dobre vidíme aj zelené stĺpce! Pripomíname, že zelené stĺpce značia NEsignifikantné hodnoty PACF pre daný lag. Na naše prekvapenie `MT` až dvakrát zlyhalo. V tomto prípade máme jednoduchý nástroj na rozsúdenie – Ljung-Box test, ktorého výsledky vidíme v Tabuľke 13.

Ljung-Box test usúdil, že aj `MT` aj `WH` mali dostatočne malé autokorelácie až po lag 17 na to, aby boli vyhodnotené za nesignifikantné. Spomedzi GPT4 datasetov obstál najlepšie `L2mod`, čo sme aj očakávali, dokonca dve parciálne autokorelácie vyšli ako nesignifikantné.



Obr. 25: Histogramy globálnej PACF pre datasety `L1` – `MT`.



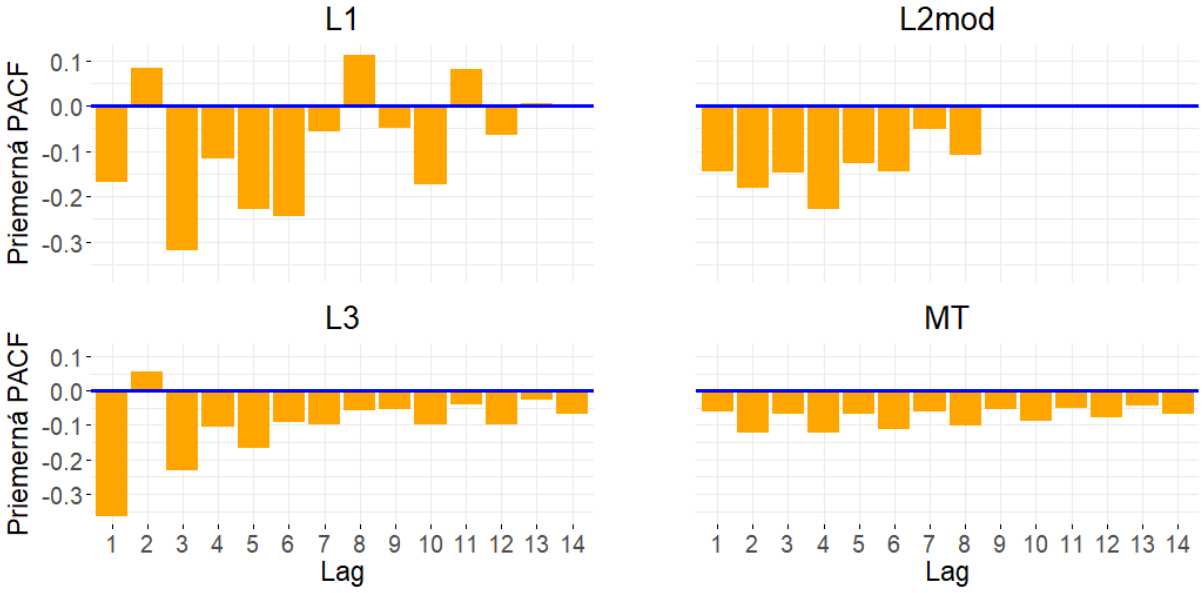
Obr. 26: Histogramy globálnej PACF pre datasety `MT` a `WH`.

Pozrime sa teraz na výsledky parciálnych autokorelácií v rámci čísel zobrazené na Obrázku 27. Opäť pripomíname, že tento odhad je biasovaný (aj potvrdené literatúrou [24]). V tejto vizualizácii je vychýlenie odhadu jasne badateľné pre datasety `MT` a `WH`.

`L2mod` je dramaticky iný, skoro všetky hodnoty PACF sú silno negatívne. Toto správanie vieme vysvetliť práve vychýlením odhadu pre malé n. V prípade `L2mod` je n ešte menšie než v ostatných prípadoch, keďže veľká časť čísel v tomto datasete má dĺžku iba

Dataset	Použitý test	n_cifier	Hodnota štatistiky	df	p-value	n_PASS	Lag
`L1`	Ljung-Box	320000	$Q = 136855,79$	17	0,0000	1/17	13
`L2mod`	Ljung-Box	216911	$Q = 3339,64$	17	0,0000	2/17	6,8
`L3`	Ljung-Box	320000	$Q = 119094,07$	17	0,0000	0/17	-
`MT`	Ljung-Box	320000	$Q = 23,58$	17	0,1312	15/17	ALL - 6,10
`WH`	Ljung-Box	320000	$Q = 7,04$	17	0,9829	17/17	ALL

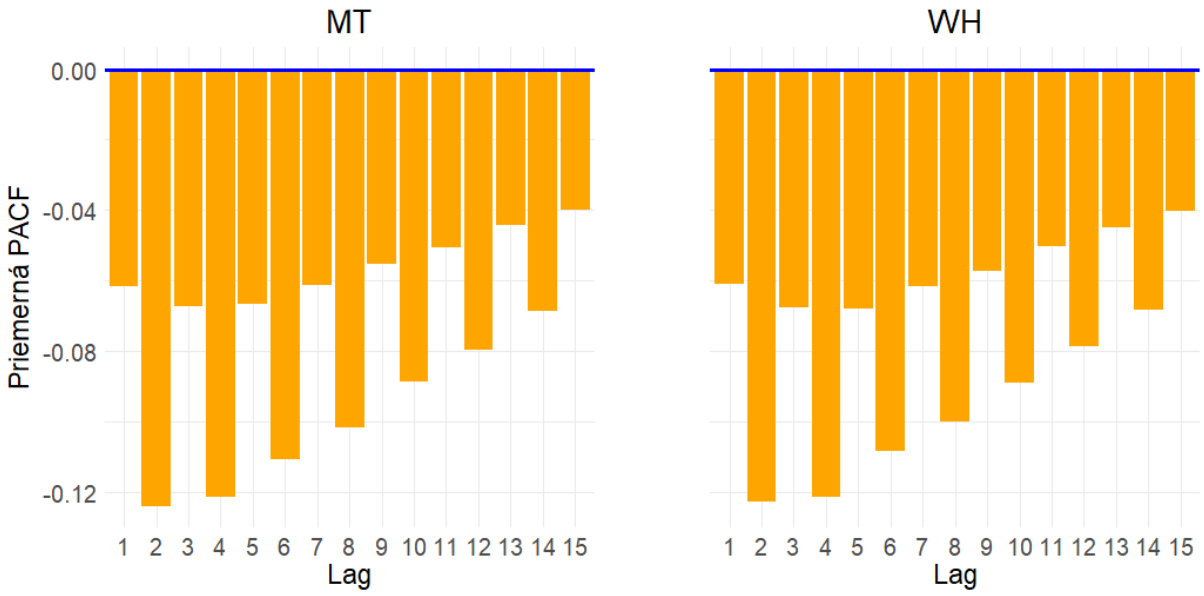
Tabuľka 13: Výsledky Testu 6.



Obr. 27: Histogramy priemernej PACF v rámci čísel pre datasety `L1` – `MT`.

9 namiesto 16, čo vedie práve ku zväčšeniu výchylky odhadu.

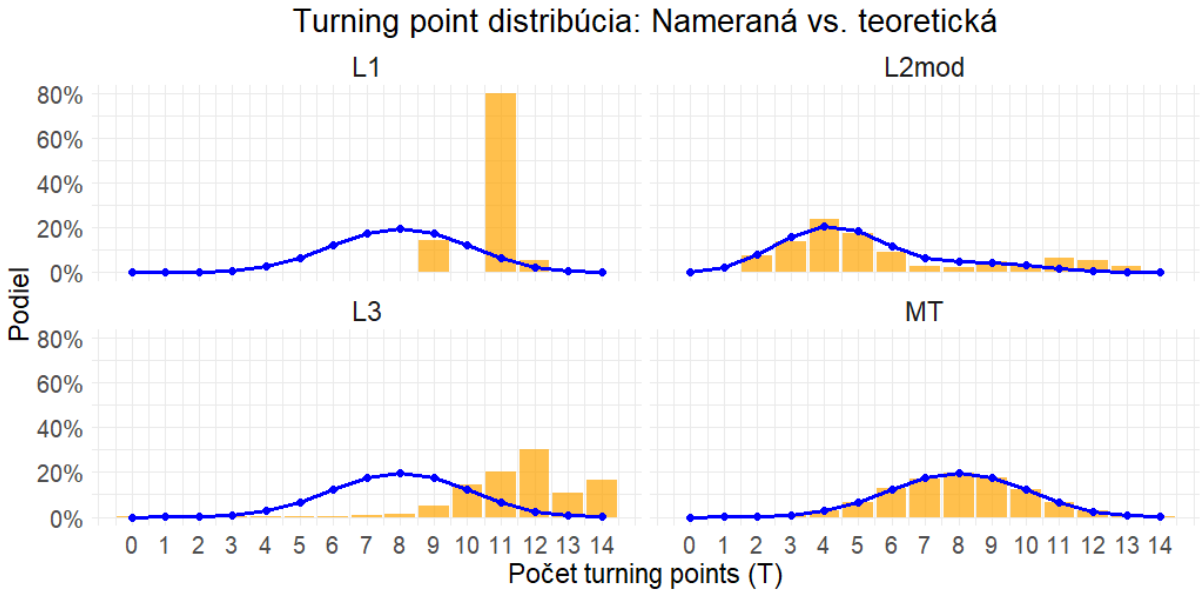
Na Obrázku 28 vidíme porovnanie `MT` a `WH`, no v skutočnosti je na ňom dobre vidieť vychýlenie odhadu PACF pre malé n, u nás nanajvýš 16. Táto výchylka sa s rastúcim n znižuje a empirický odhad PACF je asymptoticky nevychýlený (preto vychádza dobre pre $n = 20000$ vyššie).



Obr. 28: Histogramy priemernej PACF v rámci čísel pre datasety `MT` a `WH`.

- **Otočné body v rámci čísel (Turning-Point Test) (Test 7)**

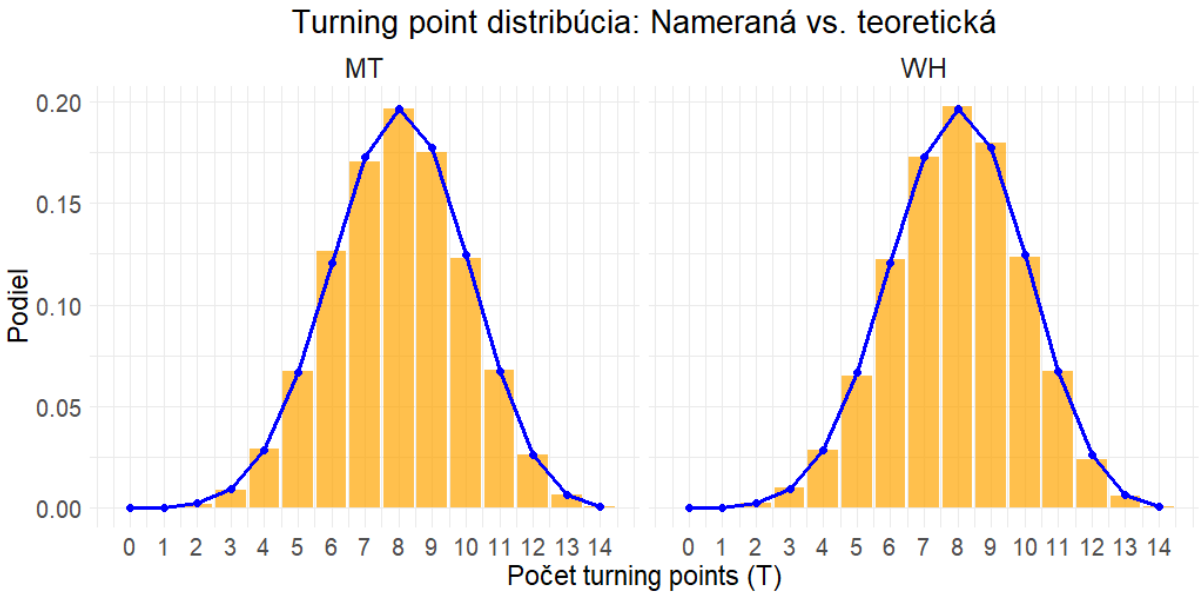
Na Obrázku 29 môžeme vidieť vizualizáciu početností otočných bodov v datasetoch `L1` až `MT`. `L1` a `L3` opäť majú veľmi podobnú povahu, oba rámce obsahujú priveľa otočných bodov. `L2mod` vyzerá v tomto ohľade omnoho lepšie, no stále vidno, že pravý chvost rozdelenia je stále príliš ťažký.



Obr. 29: Pozorované distribúcie počtu otočných bodov pre datasety `L1` – `WH`.

Pozrime sa na Obrázok 30, kde sú priamo zobrazené rámce `MT` a `WH`. Rozdiely medzi

MT a WH vyzerajú byť veľmi malé. Obe sú veľmi blízke ku očakávanej distribúcii, čo nám aj istým spôsobom utvrdzuje správnosť výpočtu teoretickej distribúcie pomocou dynamického programovania.



Obr. 30: Pozorované distribúcie počtu otočných bodov pre datasety MT a WH.

V Tabuľke 14 vidíme výsledky testu otočných bodov pre všetky datasety. L2mod je znova najlepší spomedzi rámcov tvorených pomocou GPT4, s rádovo nižšou hodnotou testovej štatistiky. Test prešli však iba pseudonáhodne generované čísla.

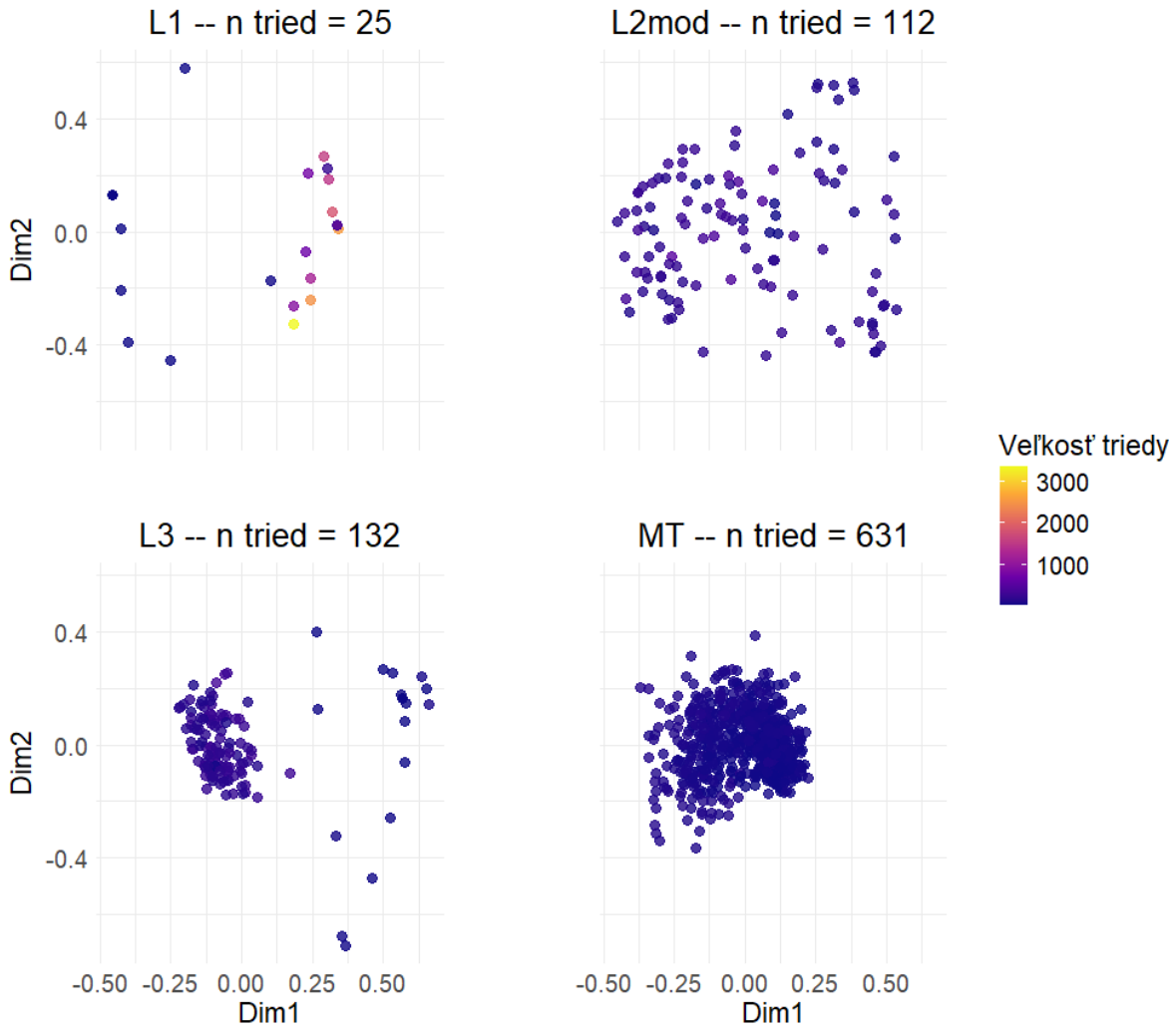
Dataset	Použitý test	Hodnota štatistiky	df	Pomer	p-value	PASS/FAIL
L1	Chi-squared	$X^2 = 132780,10$	13	-	0,0000	FAIL
L2mod	Chi-squared	$X^2 = 17110,07$	13	0,736	0,0000	FAIL
L3	Chi-squared	$X^2 = 773698,70$	13	-	0,0000	FAIL
MT	Chi-squared	$X^2 = 13,15$	13	-	0,4361	PASS
WH	Chi-squared	$X^2 = 10,60$	13	-	0,6446	PASS

Tabuľka 14: Výsledky Testu 7.

- **Rozdeľovanie čísel do tried**

Na Obrázku 31 môžeme vidieť MDS graf rozdelenia do tried pre datasety `L1` až `MT`. Vidíme značné rozdiely medzi rámcami generovanými GPT4. Pre dataset `L1` bolo detegovaných iba 25 tried, čo sedí s našimi očakávaniami. `L1` preukazoval vyslovene opakovacie správanie.

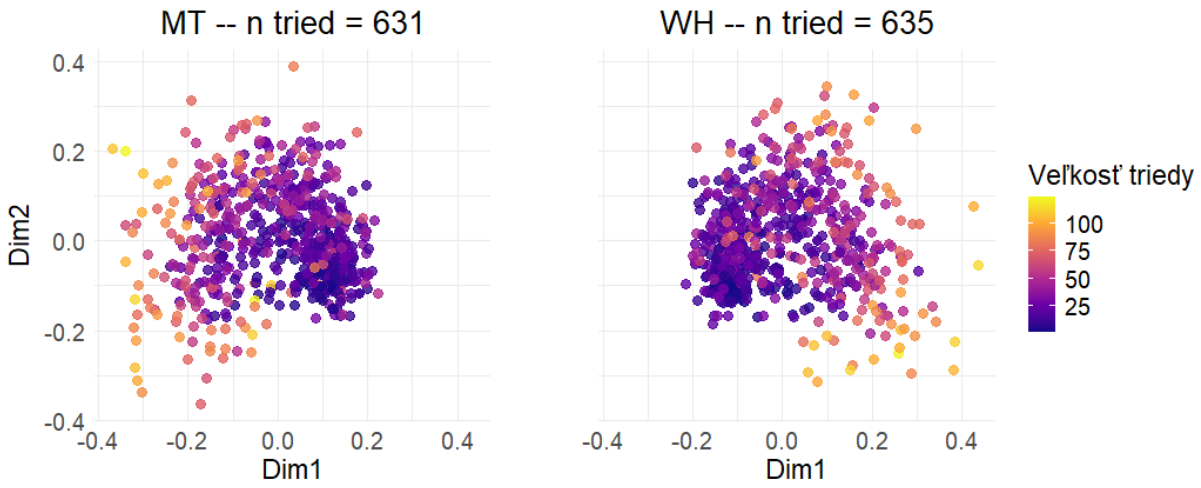
`L2mod` už má tried omnoho viac a to 112. Triedy nepreukazujú žiadne ďalšie výrazné známky zhlukovania. `L3` má tried opäť o kúsok viac, 132. Vyzerá to tak, že vznikol jeden primárny zhluk a potom viac tried, ktoré nemajú veľkú podobnosť s ostatnými ďalekými triedami, či zhlukom. Pozrieme sa na to bližšie v dendrograme neskôr.



Obr. 31: MDS vizualizácia pre `L1` – `MT`.

`MT` nepreukazuje žiadne známky zhlukovania a má významne viac tried, až 631. Bližšie ho vidíme v porovnaní s `WH` na Obrázku 32. `MT` a `WH` vyzerajú veľmi podobne. Ich triedy s vysokou kardinalitou sú si navzájom aj podobnejšie. Túto skutočnosť sme si chceli

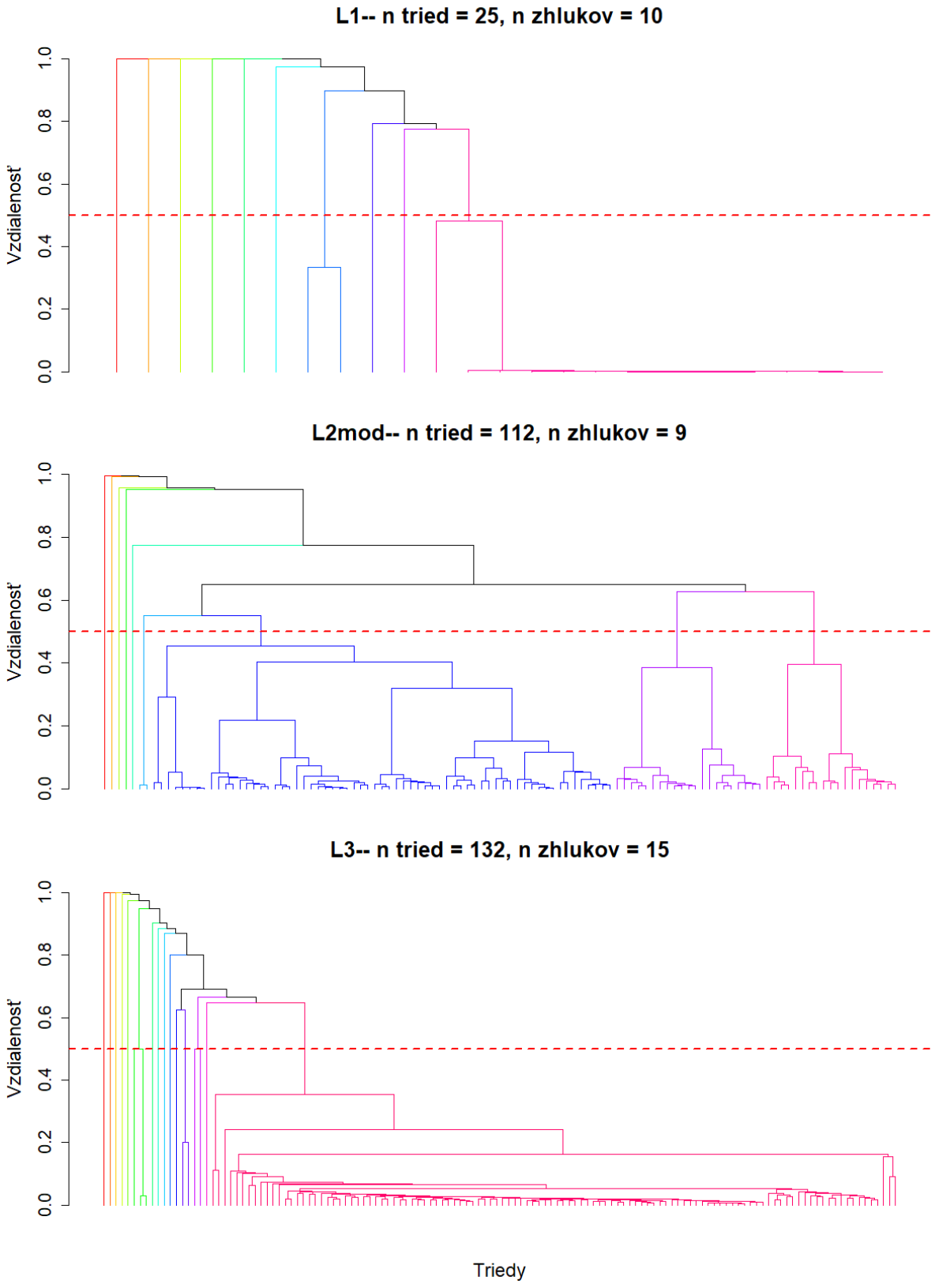
preveriť aj v dátach. Kardinalita tried postupne klesá s číslom triedy (čím neskôr bola trieda vytvorená, tým má nižšiu kardinalitu), tak ako by sme očakávali pre náhodné čísla.



Obr. 32: MDS vizualizácia pre `MT` a `WH`.

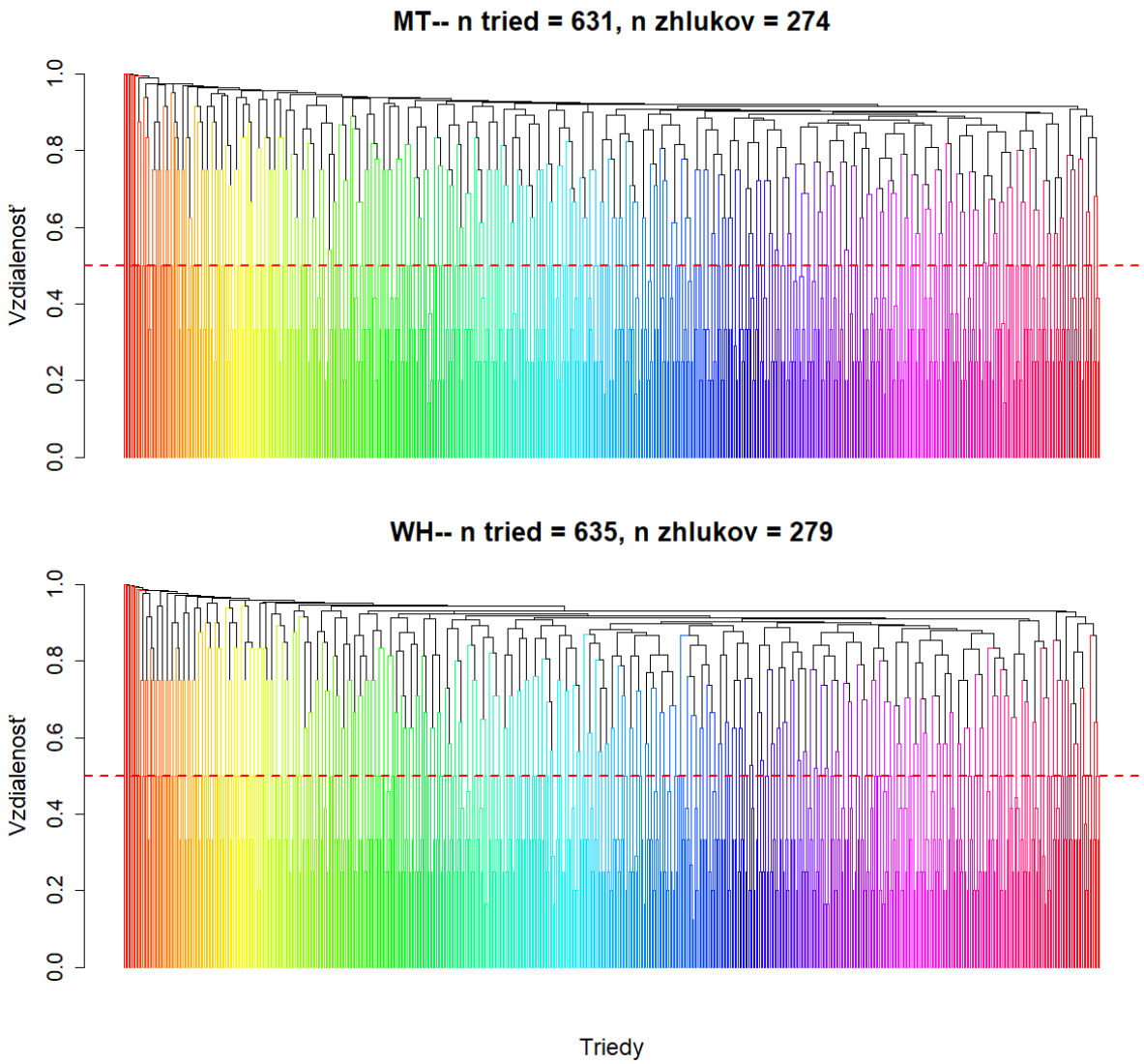
Na Obrázkoch 33 a 34 môžeme vidieť dendrogramy rozdelení do tried pre datasety `L1` až `L3` a potom `MT` a `WH`. Celkovo sa datasety `L1` až `L3` dajú v tomto charakterizovať malým alebo stredným počtom tried a veľmi malým počtom zhlukov. Naopak rámce `MT` a `WH` sa dajú charakterizovať veľkým počtom tried a veľkým počtom zhlukov. Rámce `MT` a `WH` majú veľmi podobné počty tried aj zhlukov.

Dendrogram pre L1 – L3



Obr. 33: Dendrogram pre L1 – L3.

Dendrogram pre `MT` a `WH`



Obr. 34: Dendrogram pre `MT` a `WH`.

3.2 Implicitné testovanie generovaných postupností

V tejto kapitole budeme experimentovať s nami generovanými číslami menej priamo – budeme ich využívať ako zdroje náhodnosti v optimalizačných algoritmoch. Pozrime sa na pozorované výsledky.

3.2.1 Simulované žíhanie (SA)

Nastavenia algoritmu a predpisy použitých funkcií nájdeme v Tabuľke 1 v predchádzajúcej kapitole. Vizualizácie sú zoradené podľa jednotlivých funkcií. Každá vizualizácia sa skladá z dvoch stĺpcov, pre dve použité techniky extrahovania náhodnosti z datasetov.

Prvý riadok vizualizácie predstavuje scenár s absolútnou hranicou konvergencie ε, konkrétne pre

$$\varepsilon \in \{10,\ 1,\ 10^{-1},\ 10^{-2},\ 10^{-3}\}.$$

Druhý riadok vizualizácie predstavuje scenár s relatívnou hranicou konvergencie, konkrétne pre

$$\delta_k = f(x^{(0)})\left(\tfrac{1}{2}\right)^k, \quad k = 1, 2, \ldots, 8.$$

kde os x je práve v jednotkách δ_k.

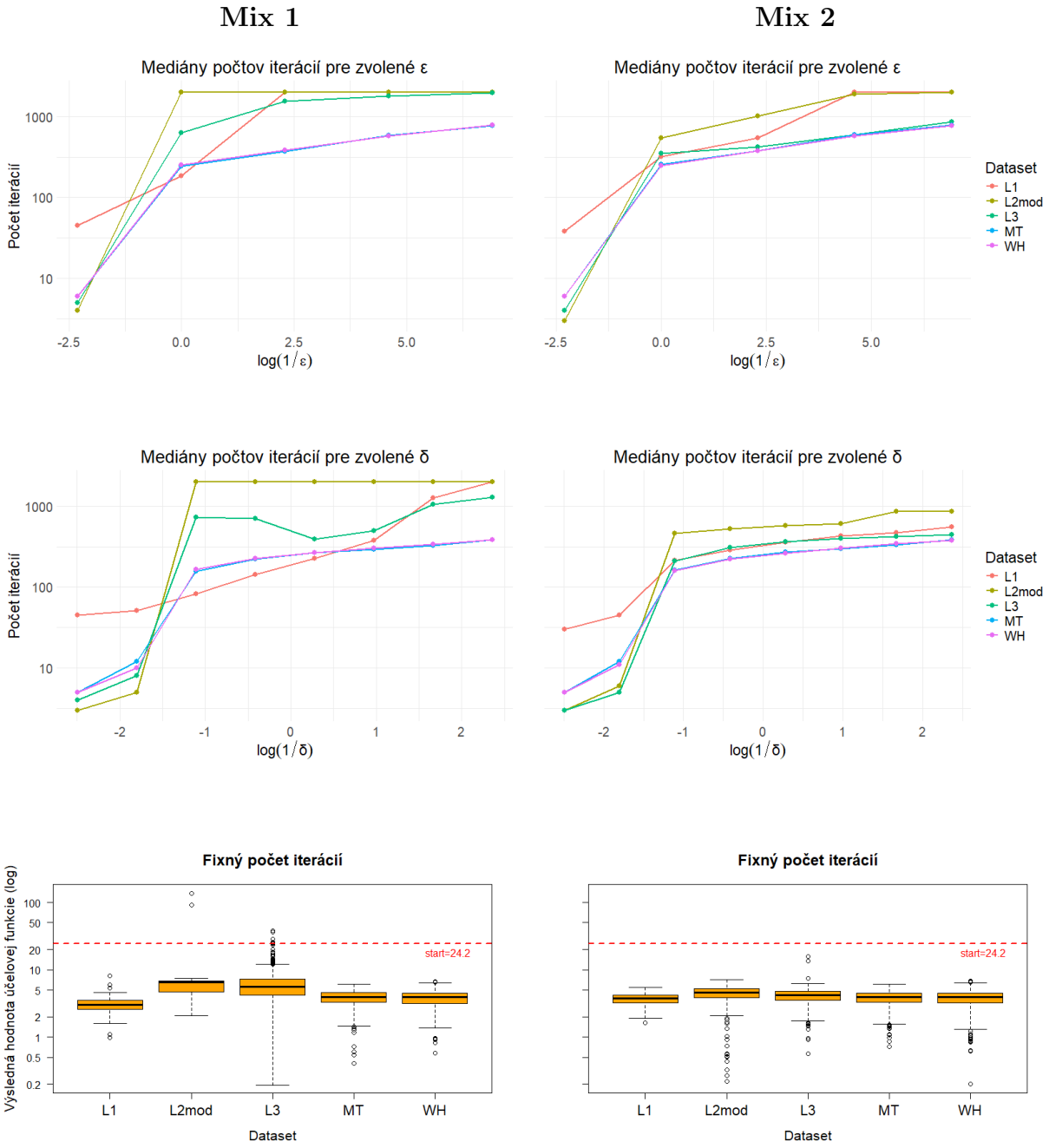
Posledný riadok vizualizácie je scenár `iter_cap`, kde každá zo simulácií bežala fixný počet iterácií, konkrétne 100. Zobrazenie je prevedené pomocou boxplotu, nech si vieme dobre predstaviť aj variabilitu v rámci simulácií. Širší box znamená väčšiu vzdialenosť medzi prvým a tretím kvartilom, čo značí väčšiu variabilitu výsledných hodnôt simulácií. Červenou prerušovanou čiarou je zaznačená hodnota danej funkcie v začiatočnom bode optimalizácie.

Keď už máme vizualizácie vysvetlené, pozrime sa postupne na Obrázky 35, 36, 37 a 38.

• **Rosenbrock: Obrázok 35**

→ **Absolútna ε konvergencia:** Mix 2 dokázal značne zlepšiť výkon datasetov `L1` až `L3`. `L3` sa v ňom dokázal vyrovnať pseudonáhodným číslam a `L2mod` skonvergoval pre viac hodnôt ε. V Mix 1 dataset `L1` aj porazil pseudonáhodné čísla pre $\varepsilon = 1$, no len o málo.

→ **Relatívna δ konvergencia:** V Mix 1 bol pre 3 rôzne hodnoty δ najlepší práve `L1`, toto prvenstvo však zmizne pri Mix 2. Mix 2 však dokázal znížiť nutné počty iterácií pre zvyšné rámce generované GPT4.



Obr. 35: Simulated annealing aplikovaný na funkcii Rosenbrock pre dve techniky miešania.

→ **Iteračne limitovaný scenár:** Aj v Mix 1 aj v Mix 2 bol najlepší dataset `L1`, v Mix 1 bol dokonca jeho tretí kvantil nižšie ako prvý kvantil pre pseudonáhodné čísla. Mix 2 všetky datasety priniesol na podobnú úroveň výslednej hodnoty účelovej funkcie po 100 iteráciach.

- **Rastrigin: Obrázok 36**

→ **Absolútna ε konvergencia:** Aj Mix 1 a Mix 2 preukazujú najnižší počet iterácií pre $\varepsilon = 10$ pri datasete `L1`. Pre ďalšie ε už však buď datasety neskonvergovali, alebo skonvergovali iba pseudonáhodné.

→ **Relatívna δ konvergencia:** Mix 2 opäť dokázal zlepšiť efektivitu datasetov generovaných GPT4. Opäť sa dataset `L1` javí ako veľmi dobrý na začiatku ale potom sa dramaticky zhorší. Pseudonáhodné čísla si zachovávajú rovnakú rýchlosť konvergencie nezávislo od štádia, v ktorom sa nachádzajú.

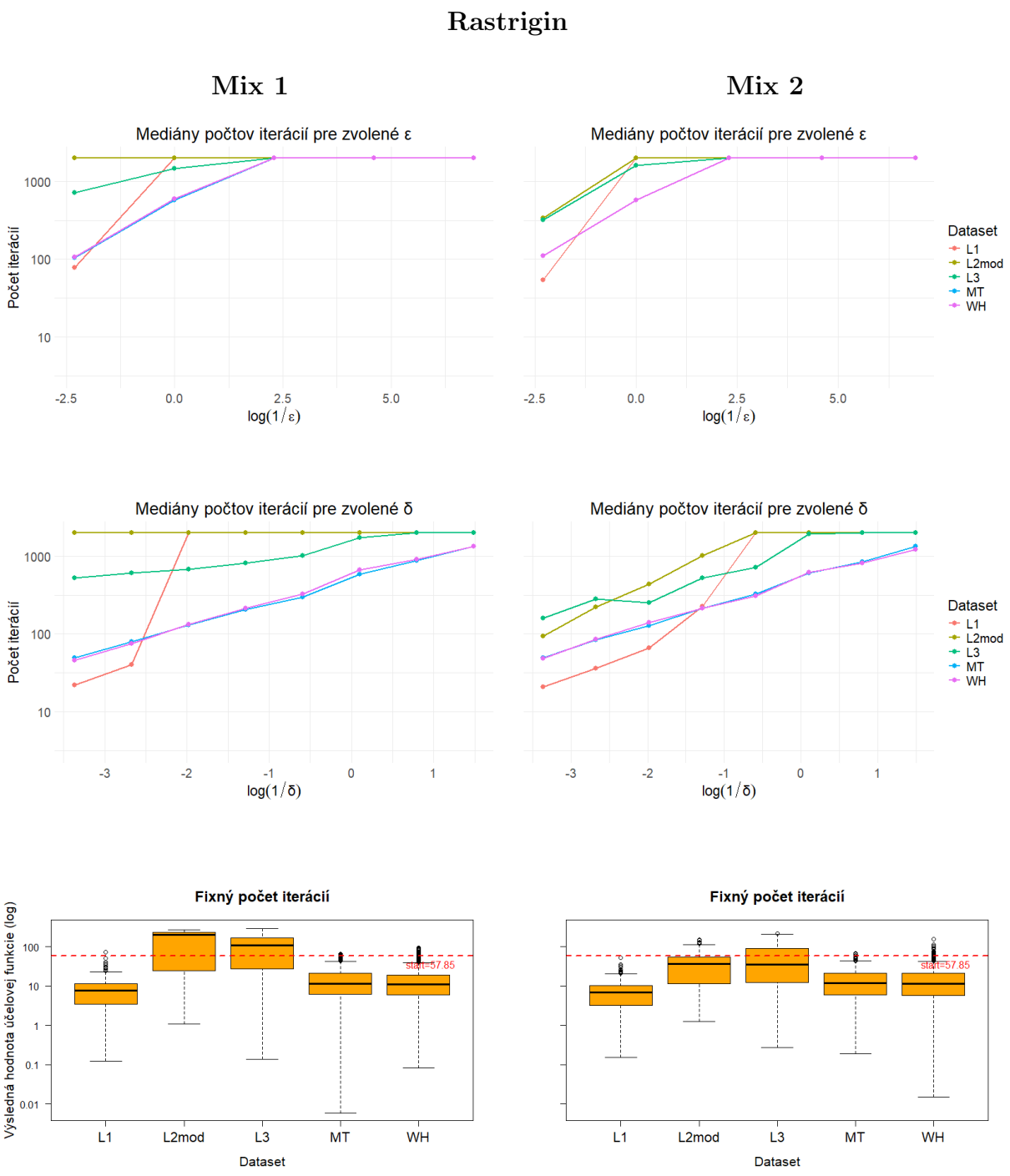
→ **Iteračne limitovaný scenár:** `L1` opäť vychádza najlepšie po 100 iteráciách, aj varianciu má relatívne nízku. Mix 2 dokázal pomôcť datasetom `L2mod` a `L3`, no stále často ani po 100 iteráciach nedokázali dosiahnuť lepšiu výslednú hodnotu ako začiatočnú.

- **Griewank: Obrázok 37**

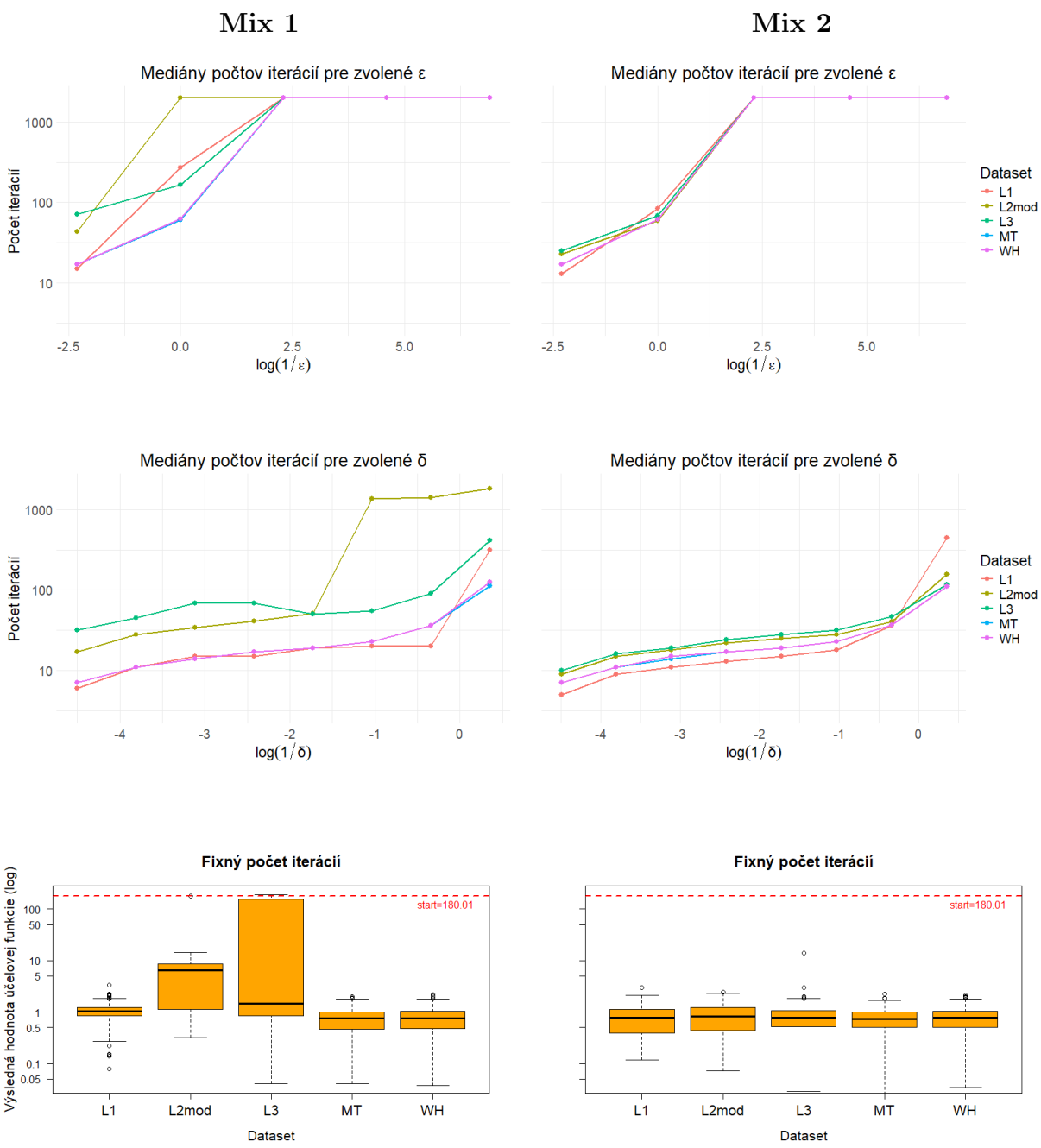
→ **Absolútna ε konvergencia:** Mix 2 opäť dokázal priblížiť výkon všetkých GPT4 datasetov k pseudonáhodným. Pre prvú hodnotu ε opäť vyhráva `L1`. Pre nižšie hodnoty ε nestačil maximálny počet iterácií pre žiadny dataset.

→ **Relatívna δ konvergencia:** Pre rôzne δ sa držia pokope datasety `L1`, `MT` a `WH`. Mix 2 dokázal dokonca vytiahnuť `L1` na prvé miesto spomedzi všetkých datasetov na dlhú dobu. Je dôležité však upozorniť na nízky absolútny počet iterácií, v Mix 2 sa celú dobu pohybujeme pod 100 iteráciami.

→ **Iteračne limitovaný scenár:** Pre Mix 1 mal dataset `L3` občas potiaže dostať sa vôbec pod začiatočnú hodnotu účelovej funkcie. Mix 2 dokázal opäť vyrovnať rozdiely medzi datasetmi. Určite chceme poukázať na nízku varianciu datasetu `L1`.



Obr. 36: Simulated annealing aplikovaný na funkcii Rastrigin pre dve techniky miešania.



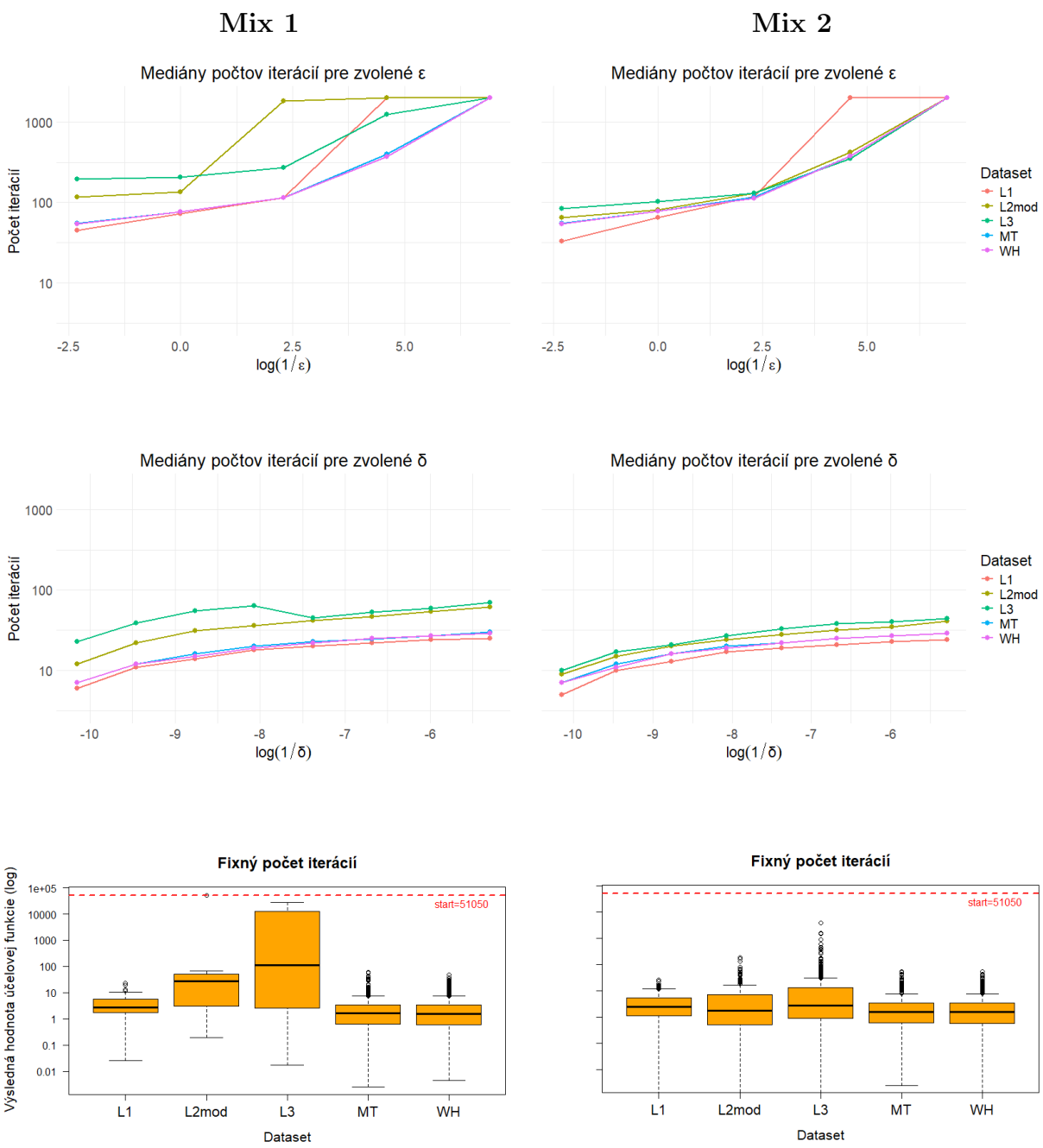
Obr. 37: Simulated annealing aplikovaný na funkcii Griewank pre dve techniky miešania.

- **Zakharov: Obrázok 38**

→ **Absolútna ε konvergencia:** Vidíme tu veľmi podobné vzory ako v predošlých funkciách. Mix 2 opäť dorovnáva rozdiely, `L1` je najlepší pre veľké hodnoty ε.

→ **Relatívna δ konvergencia:** Dataset `L1` vyhráva nielen pre Mix 1 ale aj pre Mix 2, no rozdiely sú veľmi malé. Dataset `L3` ukazoval už aj v iných prípadoch nemonotónne správanie, podľa všetkého je tak z dôvodu vysokej variancie výsledkov pre tento dataset.

→ **Iteračne limitovaný scenár:** Mix 2 dramaticky znížil variancie výsledkov datasetov `L3` a `L2mod`. `L1` si udržuje stabilne dobré výsledky v tomto scenári.



Obr. 38: Simulated annealing aplikovaný na funkcii Zakharov pre dve techniky miešania.

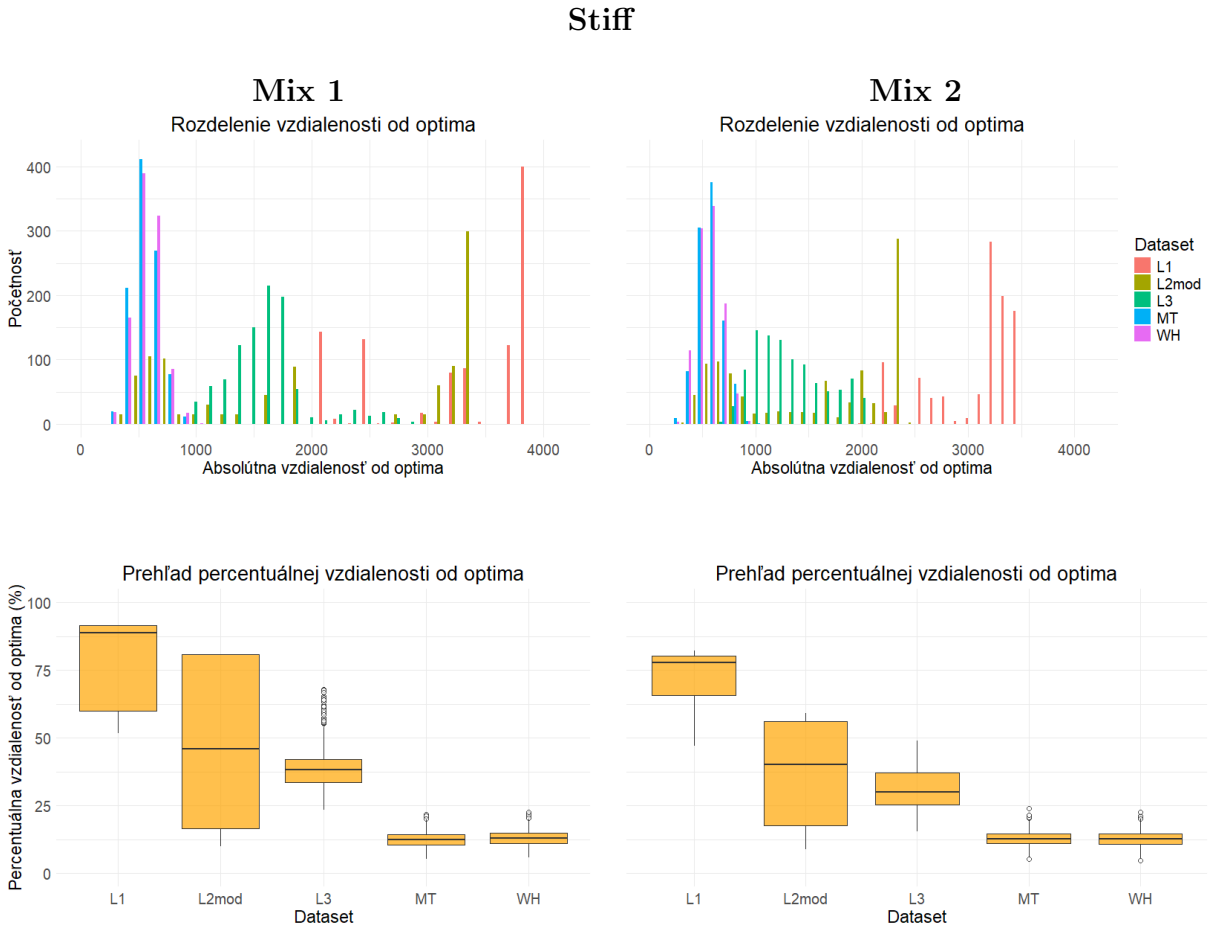
3.2.2 Knapsack

Vizualizácie pre Knapsack (Obrázky 39 a 40) sa opäť skladajú z dvoch stĺpcov, kde každý stĺpec predstavuje iný typ extrahovania náhodnosti z našich datasetov.

Vizualizácie sú rozdelené na dve podľa druhu algoritmu, Stiff a Relaxed. V prvom riadku je zobrazené rozdelenie vzdialenosti od optima, ktoré je robené formou histogramu. Celkovo sme vykonali 1000 simulácií pre každý dataset. Tieto simulácie sme následne rozdelili do binov na základe práve výslednej vzdialenosti od optima. Čím je teda distribúcia viac vľavo a jej variancia nižšia (teda stĺpce vyššie a bližšie k sebe), tým lepší je daný dataset ako zdroj pre algoritmus.

Druhý riadok je iný pohľad na tú istú metriku, je na ňom boxplot pre vzdialenosť od optima na konci algoritmu, len je to tentoraz percentuálna vzdialenosť.

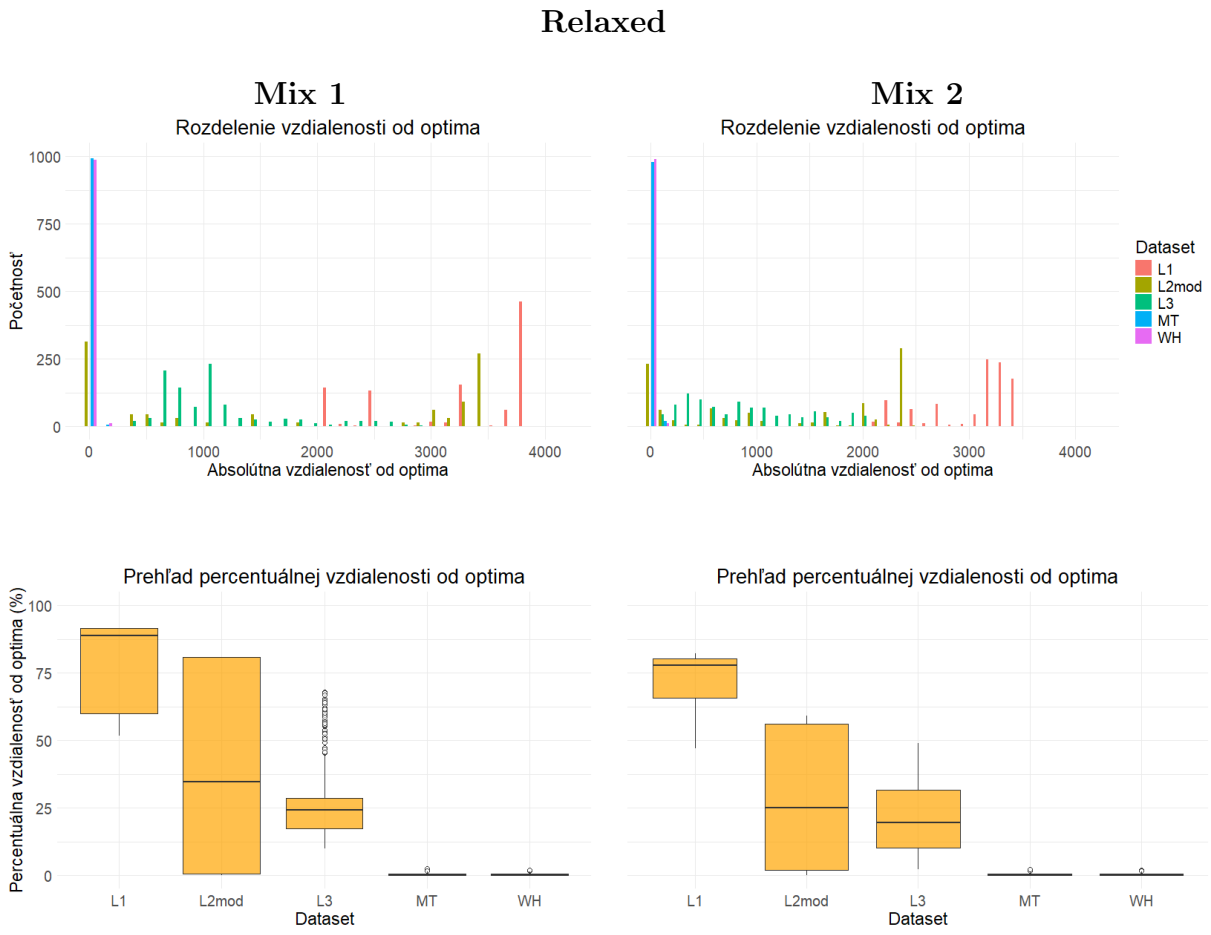
- **Stiff Knapsack: Obrázok 39**



Obr. 39: Riešenia problému Knapsack optimalizovaného pomocou SA – Stiff verzia

→ **Rozdelenie vzdialenosti od optima:** Vidíme najlepšie výsledky pre datasety `MT` a `WH`. Zaujímavé bolo pre nás správanie `L2mod`, malo jednu plynulejšiu časť svojho rozdelenia okolo hodnoty 600, no potom malo jeden vysoký stĺpec ďalej vpravo (pre Mix 1 pri hodnote 3300 a pre Mix 2 pri hodnote 2300). Ťažko povedať, ktorý z LLM datasetov bol najlepší v tomto experimente. `L3` bol asi najviac konzistentný, ako jediný nemal extrémne vysoké stĺpce mimo hlavného tela svojho rozdelenia. Mix 2 opäť dokázal zlepšiť výkon datasetov `L1` až `L3`. Na datasety `MT` a `WH` nemala zmena miešania významný efekt.
→ **Boxplot percentuálnej vzdialenosti od optima:** Mix 2 dokázal dramaticky znížiť varianciu datasetov `L1` až `L3`, čo je veľmi vítaný efekt. Mix 2 znížil aj hodnotu mediánu ich percentuálnej vzdialenosti od optima. Na výsledky `MT` a `WH` nemala zmena miešania významný efekt. Najlepšie určite vychádzajú pseudonáhodné datasety.

- **Relaxed Knapsack: Obrázok 40**



Obr. 40: Riešenia problému Knapsack optimalizovaného pomocou SA – Relaxed verzia

→ **Rozdelenie vzdialenosti od optima:** Vidíme najlepšie výsledky pre datasety `MT` a `WH`, drvivá väčšina simulácií dosiahla optimálnu hodnotu vypočítanú dynamickým programovaním. Mix 2 opäť prilepšil všetkým LLM datasetom.
→ **Boxplot percentuálnej vzdialenosti od optima:** Mix 2 dokázal dramaticky znížiť varianciu všetkých datasetov až na `L3`. Relaxácia algoritmu viedla k drasticky lepším výsledkom, no najmä pri `MT` a `WH`, ktoré veľmi často dosiahli optimálnu hodnotu.

3.3 Diskusia k výsledkom

V tejto podkapitole rozoberieme získané výsledky z explicitného aj implicitného testovania (Podkapitoly 3.1 a 3.2), navrhneme hypotézy o vlastnostiach generovaných čísel, ktoré môžu vysvetliť pozorované správanie a poukážeme na možnosti zlepšenia kvality prostredníctvom pokročilého promptovania a techník extrakcie náhodnosti z generovaných sekvencií.

Zistenia z explicitného testovania

Explicitné Testy (1 až 7) odhalili systematické odchýlky od ideálneho náhodného správania pre datasety generované GPT4, čo sme aj predpokladali v Podkapitole 1.2. Každý štatistický test vyvrátil skúmanú vlastnosť náhodnosti pre každý z datasetov `L1`, `L2mod` a `L3`, tento výsledok možno vidieť aj v Tabuľke 15. Dataset `L3` mal najlepšie výsledky spomedzi GPT4 rámcov v Testoch 1, 3, 4 a potenciálne pri experimente so zaradením do tried, z hľadiska počtu tried a ich kardinalít. V Testoch 2, 5, 6, 7 najlepšie obstál dataset `L2mod`, aj v experimente so zaraďovaním do tried mal dobré výsledky, kardinalita tried bola privysoká a príliš podobná medzi triedami. Dátový rámec `L1` v našom testovaní vykazoval veľmi slabé znaky náhodnosti.

Z týchto výsledkov možno zhodnotiť, že správne promptovanie má výrazný kladný efekt pri úlohe generovania náhodných čísel s konkrétnou špecifikáciou. Najväčší efekt je povšimnuteľný na udržateľnosti kvalitnej konverzácie aj pri veľkom množstve kontextu (dlhé konverzácie) a splnení základných vlastností náhodnosti (ako uniformita na danom intervale alebo uniformita cifier naprieč pozíciami). Inherentnú nenáhodnosť LLM dokázalo správne promptovanie zjemniť, no nie úplne eliminovať. Jednou takouto vlastnosťou je nevôľa využívať číslicu 6 (Obrázok 19, Test 2), ďalšou je inklinácia LLM striedať nízke cifry väčšími a vice versa (Obrázky 23 a 29, Testy 5 a 7).

Test\Dataset	L1	L2mod	L3	MT	WH
Uniformita na $[0; 1]$	×	×	×	✓	✓
Uniformita globálneho rozdelenia cifier	×	×	×	✓	✓
Uniformita cifier naprieč pozíciami	×	×	×	✓	✓
Uniformita pozícií naprieč ciframi	×	×	×	✓*	✓
Uniformita dvojíc cifier	×	×	×	✓	✓
Absencia autokorelácie	×	×	×	✓*	✓
Test otočných bodov	×	×	×	✓	✓

Tabuľka 15: Sumár výsledkov explicitného testovania pri spoľahlivosti 95%.

Dataset `L2mod` v niektorých testoch výrazne porazil rámec `L3`, i keď `L3` trval rádovo dlhšie na vytvorenie a od užívateľa vyžadoval neustálu pozornosť a analyzovanie generovaných čísel. Naša domnienka je, že tento výsledok mohol byť spôsobený práve výskytom anomálie v `L2`. Keď GPT4 generoval čísla, ktoré mali dĺžku iba deväť cifier a na ich koniec pridal sedem núl, tak sa domnievame, že všetok svoj výkon mohol alokovať do týchto deväť cifier a následne si "oddýchnuť" pred ďalším podaním výkonu. Táto domnienka vychádza aj z Obrázku 29, kde možno vidieť pre `L2mod` ťažký pravý chvost. Tento bias je viditeľný aj pre rámce `L1` a `L3`, no v `L2mod` je výsledná pozorovaná distribúcia kombináciou dvoch, jednou pre čísla s anomáliou a druhou bez. Ľavý chvost má zďaleka menšie problémy dosiahnuť očakávanú distribúciu ako pravý.

Zistenia z implicitného testovania

Pri využití generovaných čísel v optimalizačných algoritmoch (Simulované žíhanie a Knapsack) (Obrázky 35 až 40) sme zistili, že aj napriek explicitným odchýlkam môžu niektoré štruktúry dát pozitívne ovplyvniť rýchlosť konvergencie. Stratégia `L1` so silným opakujúcim sa vzorom priniesla signifikantné zlepšenie v počiatočných iteráciách SA oproti pseudonáhodným datasetom `MT` a `WH`, no neskôr stagnovala. Datasetom `L2mod` a `L3` sa nám podarilo zlepšiť výkon aj konzistentnosť pomocou inej stratégie extrahovania náhodnosti aj v konvergenčne aj iteračne limitovaných scenároch. V experimentoch Knapsack Mix 2 znížil variabilitu aj medián percentuálnej vzdialenosti od optima, pričom pseudonáhodné datasety zostali najspoľahlivejšie. Ukázali sme aj, že rôzne miešacie techniky na extrakciu

čísel, ktoré následne slúžia ako zdroj náhody pre optimalizačné algoritmy, majú významný vplyv na výkon čísel generovaných LLM v tomto kontexte.

Ako prvý možný dôvod dobrého výkonu `L1` pri optimalizácii spojitej funkcie pomocou SA sme ako prvé chceli viac vylúčiť náhodu. Experimentovali sme teda s funkciou Rosenbrock, ktorá nie je symetrická, tým spôsobom, že sme iba vymenili poradie premenných (ako vo funkcii Rosenbrock, tak aj v začiatočnom bode) a následne sledovali výkon všetkých skúmaných datasetov. Ak by bol výkon porovnateľný, môžeme tento faktor vylúčiť. `L1` si udržalo prvé miesto v iteračne limitovanom scenári aj po tejto zmene, výkon `L2mod` a `L3` sa v tomto scenári zhoršil a výkon pseudonáhodných datasetov ostal porovnateľný. Domnievame sa, že vlastnosť rámca `L1`, ktorá spôsobila tento dobrý výkon na začiatku iterovania je práve opakovanie čísel. Pri nižšej hranici konvergencie (menšie ε) však `L1` začal zaostávať za pseudonáhodným zdrojom náhody. Preto potenciálny postup na zrýchlenie optimalizačných algoritmov by mohol nasledovať nasledovnú štruktúru: Najskôr by bolo použité opakovanie ako v `L1` a následne by sa použité čísla preklopili na pseudonáhodné, keď je už dosiahnutá istá vzdialenosť od optima.

Záver

Táto práca sa zaoberala analýzou techník efektívnej interakcie s veľkými jazykovými modelmi (LLM), pričom ako experimentálnu úlohu sme zvolili generovanie náhodných čísel. Prvým cieľom bolo otestovať kvalitu náhodnosti týchto čísel, druhým preveriť a zdokumentovať rôzne prístupy k promptovaniu a práci s LLM. Generovanie náhodných čísel tak slúžilo nielen na hodnotenie ich štatistických vlastností, ale aj na odhalenie možností a obmedzení súčasných modelov a demonštráciu osvedčených spôsobov ich využitia.

Práca začala analýzou histórie a vývoja LLM v Kapitole 1. Toto teoretické zázemie bolo nevyhnutné na pochopenie správania LLM pri riešení úloh mimo ich primárneho účelu. V Podkapitole 1.2 sme definovali náhodnosť, pomenovali rozdiely medzi skutočnou náhodnosťou a pseudonáhodnosťou a spísali naše očakávania o výkone LLM v tejto úlohe. Definícia náhodnosti bola kľúčová pre návrh testovacieho balíka, ktorý pozostával z explicitných a implicitných testov (Kapitola 2). Použité explicitné testy umožnili presnú analýzu štatistických vlastností generovaných dát, zatiaľ čo implicitné testy odhalili, ako náhodnosť ovplyvňuje ďalšie procesy. Kombináciou týchto testov sme získali objektívny obraz o kvalite výstupov LLM.

Podkapitola 2.1 sa zamerala na proces samotného generovania náhodných čísel pomocou LLM. Tento proces odhalil, že správne prístupy, ako je precízne navrhnutý prompt či vstúpenie do "roleplay" s LLM, sú kľúčové na dosiahnutie úspechu. Iteratívne skúšanie rôznych stratégií ukázalo, že modely často vyžadujú presné usmernenie a adaptáciu prístupu na základe ich odozvy. Domnievame sa, že tieto poznatky môžu byť aplikované aj na iné úlohy, ktoré presahujú primárny účel LLM.

V Kapitole 3 sa aplikoval vytvorený testovací balík na výstupy LLM a tieto dátové rámce boli porovnané s tradičnými pseudonáhodnými generátormi, ako je Mersenne Twister. Výsledky explicitného testovania v Tabuľke 15, podrobne diskutované v Podkapitole 3.3, preukázali, že napriek zlepšeniam v promptovaní žiaden z GPT4 datasetov nedosahuje dostatočnú zhodu s očakávanou náhodnosťou. V diskusii v Podkapitole 3.3 však spomíname, že i keď LLMs nedokážu spoľahlivo generovať čísla, ktoré by vykazovali základné vlastnosti náhodnosti, je možné sa k tomuto cieľu priblížiť pomocou správneho promptovania až po limity samotnej architektúry či obmedzení vychádzajúcich zo štruktúry ich trénovacích dát. V tejto podkapitole sme tiež identifikovali aj zaujímavé

javy ako príliš časté striedanie cifier či systematicky malé zastúpenie cifry 6 v sekvenciách generovaných LLM. Nezanedbateľný pokrok oproti jednoduchému promptovaniu vo výsledkoch explicitného testovania však dosiahli stratégie `L2mod` a `L3`, podrobne vysvetlené v Podkapitole 2.1.4.

Implicitné testovanie, detailnejšie vyhodnotené v diskusii v Podkapitole 3.3, prinieslo ďalší rozmer hodnotenia čísel generovaných GPT4: využili sme generované čísla v simulovanom žíhaní a v dvoch variantoch Knapsack problému (stiff aj relaxed). Precíznejšie promptované datasety `L2mod` a `L3` v kombinácii so správnym výberom techniky extrakcie náhodnosti sme v probléme optimalizácie spojitej funkcie v Podkapitole 2.3.1 dokázali "prinútiť" vykazovať správanie blízke pseudonáhodným sekvenciám. Tieto experimenty ukázali, že rôzne spôsoby extrakcie náhodnosti z LLM výrazne ovplyvňujú správanie optimalizačných algoritmov. Rámec `L1` si však našiel uplatnenie v implicitných experimentoch. Domnievame sa, že práve vďaka explicitným nedostatkom v náhodnosti prinieslo opakovanie čísel v dátovom rámci `L1` dokonca rýchlejší úvodný prírastok kvality riešenia ako pseudonáhodné sekvencie čísel.

V práci sme ukázali, že prompt engineering dokáže čiastočne zmierniť inherentné odchýlky LLM (napríklad absenciu cifry 6 v generovaných číslach či striedanie nízkych a vysokých cifier), hoci ich úplne nevyrieši. Správne definované požiadavky v prompte (formát, presnosť vo vyjadrovaní) výrazne znižujú variabilitu a systematické skreslenia. Metodologickým prínosom tejto práce je testovací rámec, ktorý spája podrobnú štatistickú analýzu s implicitnými experimentmi v optimalizačných algoritmoch. Odporúčané postupy možno aplikovať všade tam, kde vyžadujeme spoľahlivý, presný výstup z LLM, nielen pri generovaní náhodných čísel. V budúcnosti by sme mohli rozšíriť prácu o nové modelové varianty (napríklad Grok či open-source LLM), experimentovať s rôznymi voľbami hyperparametrov v implicitnom experimentovaní, zaviesť ďalšie štatistické testy (napr. Poker test by mohol byť zaujímavý) a určite viac preskúmať našu domnienku o efektívnosti použitia opakujúcich sa sekvencií čísel na začiatku optimalizačných algoritmov. Touto prácou sme úspešne dosiahli dvojitý cieľ – systematicky zhodnotiť kvalitu náhodnosti generovanej LLM a zdokumentovať efektívne spôsoby ich promptovania – a priniesli sme poznatky, ktoré rozširujú hranice praktického využitia veľkých jazykových modelov.

Zoznam použitej literatúry

[1] Wang et. al: *History, development, and principles of large language models: an introductory survey*, AI and Ethics, Springer, 2024, 1–17, dostupné na internete (14.11.2024): https://link.springer.com/article/10.1007/s43681-024-00583-7

[2] Stipčević et. al: *True random number generators*, Open problems in mathematics and computational science, Springer, 2014, 275–315, dostupné na internete (16.11.2024): https://cetinkayakoc.net/docs/b08.pdf

[3] Matsumoto et. al: *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*, ACM Transactions on Modeling and Computer Simulation (TOMACS), ACM New York, NY, USA, 1998, 8(1), 3–30, dostupné na internete (16.11.2024): https://dl.acm.org/doi/pdf/10.1145/272991.272995

[4] Angelike et. al: *A comparative evaluation of measures to assess randomness in human-generated sequences*, Behavior Research Methods, Springer, 2024, 56(7), 7831–7848, dostupné na internete (16.11.2024): https://link.springer.com/content/pdf/10.3758/s13428-024-02456-7.pdf

[5] Brown et. al: *Language models are few-shot learners*, Advances in Neural Information Processing Systems, 2020, 33, 1877–1901, dostupné na internete (16.03.2025): https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[6] Kojima et. al: *Large language models are zero-shot reasoners*, Advances in Neural Information Processing Systems, 2022, 35, 22199–22213, dostupné na internete (16.03.2025): https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf

[7] Achiam et. al: *Gpt-4 technical report*, arXiv preprint arXiv:2303.08774, 2023, dostupné na internete (16.03.2025): https://arxiv.org/pdf/2303.08774

[8] Li et. al: *Large language models understand and can be enhanced by emotional stimuli*, arXiv preprint arXiv:2307.11760, 2023, dostupné na internete (16.03.2025): https://arxiv.org/pdf/2307.11760

[9] Yang et. al: *Large language models as optimizers. 2023*, Accessed on, 2023, 1, dostupné na internete (16.03.2025): https://arxiv.org/pdf/2309.03409

[10] Chen et. al: *Boosting Self-Efficacy and Performance of Large Language Models via Verbal Efficacy Stimulations*, arXiv preprint arXiv:2502.06669, 2025, dostupné na internete (16.03.2025): https://arxiv.org/pdf/2502.06669

[11] OpenAI: *Prompt Engineering - Ask the Model to Adopt a Persona*, 2025, dostupné na internete (16.03.2025): https://platform.openai.com/docs/guides/prompt-engineering#tactic-ask-the-model-to-adopt-a-persona

[12] Kong et. al: *Better zero-shot reasoning with role-play prompting*, arXiv preprint arXiv:2308.07702, 2023, dostupné na internete (16.03.2025): https://arxiv.org/pdf/2308.07702

[13] Zheng et. al: *Isä helpful assistant"the best role for large language models? a systematic evaluation of social roles in system prompts*, arXiv preprint arXiv:2311.10054, 2023, 8, dostupné na internete (16.03.2025): https://www.academia.edu/download/117525954/2311.10054.pdf

[14] Xu et. al: *Beyond goldfish memory: Long-term open-domain conversation*, arXiv preprint arXiv:2107.07567, 2021, dostupné na internete (16.03.2025): https://arxiv.org/pdf/2107.07567

[15] Gekhman et. al: *On the robustness of dialogue history representation in conversational question answering: a comprehensive study and a new prompt-based method*, Transactions of the Association for Computational Linguistics, MIT Press One Broadway, 12th Floor, Cambridge, Massachusetts 02142, USA, 2023, 11, 351–366, dostupné na internete (16.03.2025): https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00549/2145199/tacl_a_00549.pdf

[16] Maharana et. al: *Evaluating very long-term conversational memory of llm agents*, arXiv preprint arXiv:2402.17753, 2024, dostupné na internete (16.03.2025): https://arxiv.org/pdf/2402.17753

[17] ChatGPT konverzácia dostupná na internete (09.04.2025): https://chatgpt.com/c/67dfd790-8ba8-8002-8759-b20108ae95e8

[18] ChatGPT konverzácia dostupná na internete (09.04.2025): https://chatgpt.com/c/67decaae-15fc-8002-a76a-599ab0e3f335

[19] ChatGPT konverzácia dostupná na internete (09.04.2025): https://chatgpt.com/c/67713693-81cc-8002-bb53-6aea55d73923

[20] Pearson, Karl: *X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, Taylor & Francis, 1900, 50(302), 157–175, dostupné na internete (13.03.2025): https://doi.org/10.1080/14786440009463897

[21] Darling, Donald A.: *The kolmogorov-smirnov, cramer-von mises tests*, The Annals of Mathematical Statistics, JSTOR, 1957, 823–838, dostupné na internete (13.03.2025): https://www.jstor.org/stable/pdf/2237048.pdf?casa_token=0zQ6pwbduCkAAAAA:nLjGBi3JflwBIayDlMCk6jldvO0cUb2U435xpoZEFZOLcaqdVkNmCrt0AH9yz__X7-bfW6NnYl5wQlPmsRV0kJr7mbDNrZM_43D1eNQdgqRQoBQP1eXcW

[22] Schneider, Ivo: *Jakob Bernoulli, Ars Conjectandi (1713)*, Landmark Writings in Western Mathematics 1640-1940, Elsevier, 2005, 88–104, dostupné na internete (15.04.2025): https://www.academia.edu/download/30746336/GrattanGuinness_I_ed_Landmark_writings_in_weste.pdf#page=108

[23] TERH, Fabian: *How to solve the Knapsack Problem with dynamic programming*, Medium, 2019, dostupné na internete (17.04.2025): https://medium.com/@fabianterh/how-to-solve-the-knapsack-problem-with-dynamic-programming-eb88c706d3cf

[24] Woolrich et. al: *Temporal autocorrelation in univariate linear modeling of FMRI data*, Neuroimage, Elsevier, 2001, 14(6), 1370–1386, dostupné na internete (09.05.2025): https://pages.stat.wisc.edu/ mchung/teaching/MIA/reading/fMRI.woolrich.NI.2001.pdf

[25] Dunn, Olive Jean: *Multiple comparisons among means*, Journal of the American Statistical Association, Taylor & Francis, 1961, 56(293), 52–64, dostupné na internete (27.02.2025): https://www.jstor.org/stable/pdf/2282330?casa_token=NbWSxLi5zCsAAAAA:60Y-6lqKB3GGf1OB1VHW_GqGyeGqsizaYLtcfh5QwB_A495235N8sh_ZirOKQy8MsjE7-AqjgOEgR03lzRnnFCBEdYNUXUKbMRps3NnM_YBuaf9W9C0SMj

[26] Brockwell, Peter J., Davis, Richard A.: *Time series: theory and methods*, Springer Science & Business Media, 1991