FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

# NI-KOP

Úloha 5

Využitie simulovaného ochladzovania na riešenie
váženého 3-SAT problému

Bc. Matej Šutý
ČVUT FIT 2022

# Popis váženého 3SAT problému

Problém je zadaný ako výrok v konjunktívnej normálovej forme (CNF), kde každá klausula obsahuje práve tri premenné z premenných $\{x_1, x_2, ..., x_n\}$. Algoritmus hľadá také ohodnotenie premenných aby výrok bol pravdivý. Každá premenná $x_i$ má váhu $w_i$. Hodnota výroku je súčet váh tých premenných, ktoré boli ohodnotené na 1.
Algoritmus hľadá také ohodnotenie aby výrok bol pravdivý a hodnota výroku bola najvyššia možná.

# Popis algoritmu simulovaného ochladzovania

**Počiatočný stav**
Algoritmus najprv generuje náhodný stav - ohodnotenie každej premennej 1 alebo 0. Stav nemusí spĺňať výrok.

**Vstupné parametre**
Parametre, ktoré ovplyvňujú beh programu sú voliteľné, ich prednastavená hodnota je v zátvorke:

a. počiatočná teplota $0 < T_0 < \infty$, prednastavená hodnota sa vypočíta automaticky (pozri *Výpočet počiatočnej teploty*),
b. faktor ochladzovania $0 < k < 1$, (0.99),
c. finálna teplota $0 <= T_F < T_0$, (0.001),
d. equilibrium $0<\alpha$ (0.6), resp. počet vnútorných cyklov $I$ (0)

**Účelová funkcia**
Slúži na ohodnotenie stavu. Počet premenných vo váženom probléme 3-SAT je $n$ a počet klauzulí je $c$. V stave je každá premenná $x_i$ z ($x_1$, $x_2$, ..., $x_n$) ohodnotená na 1 alebo 0 a každá klausula $v_i$ z {$v_1$, $v_2$, ..., $v_c$} je pravdivá alebo nepravdivá podľa ohodnotenia premenných v tejto klauzuli. Nech $m$ je súčet váh premenných, ktoré sú ohodnotené na 1, a $d$ je počet splnených klauzulí. Nech $h$ je súčet váh všetkých premenných.

```
clauses_sat_ratio = d / c
weight_true_ratio = m / h
if d == c
    return weight_true_ratio
else
    return -1*(1 - clauses_sat_ratio)*weight_true_ratio
```

S vyšším počtom splnených klauzulí sa zvyšuje hodnota účelovej funkcie. Pre nesplnený výrok má záporné hodnoty a s rastúcim počtom splnených klauzulí sa hodnota blíži k nule. Vďaka tomu, že hodnota rastie pre viac ohodnotených formulí, má algoritmus možnosť vyberať lepší stav aj keď v ňom nie je splnený výrok. Zároveň je hodnota funkcie normovaná

do rozsahu [0, 1], čo znamená, že rozsah teploty nie je závislý na hodnotách váh alebo počtu klauzulí alebo premenných.

**Prehľadávanie okolia + whitebox hľadanie správneho prehľadávania**

Vo vnútornej slučke s ***I*** iteráciami (pozri *Equilibrium*) sa generujú stavy v okolí súčasného stavu. Otestoval som dva prístupy výberu ďalšieho stavu:

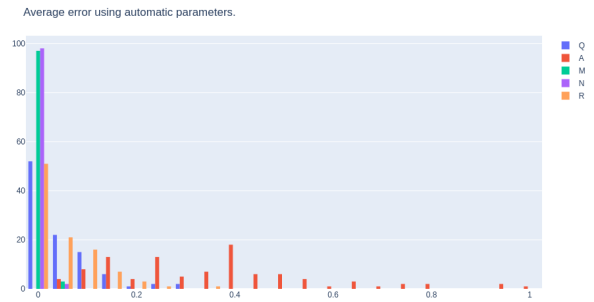
A. náhodný výber jednej premennej
B. vážený náhodný výber jednej premennej na základe váhy premennej
C. náhodný výber 1 až n/3 premenných
D. vážený náhodný výber 1 až n/3 premenných

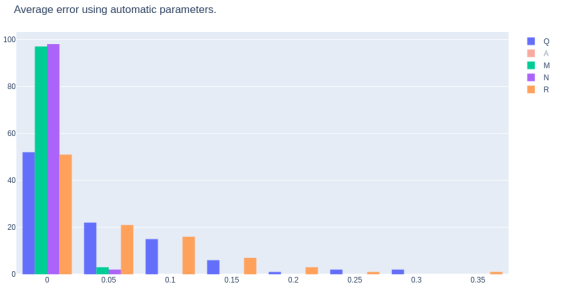
Vybranej premennej, resp. premenným, sa prevráti hodnota (1 sa zmení na 0 a opačne). Ak je tento vygenerovaný stav lepší (podľa účelovej funkcie) ako predchádzajúci stav, automaticky sa prijme a použije sa v ďalšej iterácii. Ak je hodnota účelovej funkcie stavu nižšia, algoritmus s istou pravdepodobnosťou prijme aj zhoršujúci stav. Nazvime rozdiel v hodnote účelovej funkcie ***d***. Algoritmus vygeneruje náhodné číslo ***x*** v rozmedzí **[0, 1]**. Ak je $x < exp(d / T)$, algoritmus prijme nový stav aj keď má nižšiu hodnotu účelovej funkcie.

V tabuľke nižšie je vidno spoločnú priemernú chybu na datasetoch *A, Q, M, N, R* v instanciách s 20 premennými s 78 klauzulami (pre A 88 klauzúl).

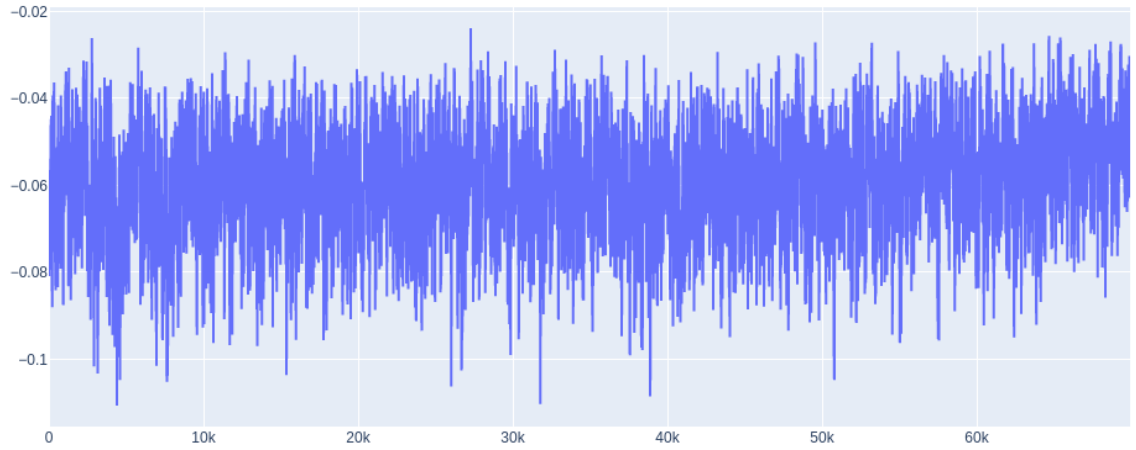
| | |
|---|---|
| A. 1 náhodná prem. | **0.9746165561565441** |
| B. 1 vážená náhodná prem. | 0.8839658779610785 |
| C. 1 - (n/3) náhodných prem. | 0.9252675549089429 |
| D. 1 - (n/3) vážených náhodných prem. | 0.9260727029950327 |

Najlepšie výsledky vykazuje použitie náhodného malého kroku pri explorácii - prevrátenie hodnoty jednej premennej. Grafy nižšie ukazujú, že algoritmus mal problém hlavne s instanciami z datasetu *A*, občas aj 100% chyba.



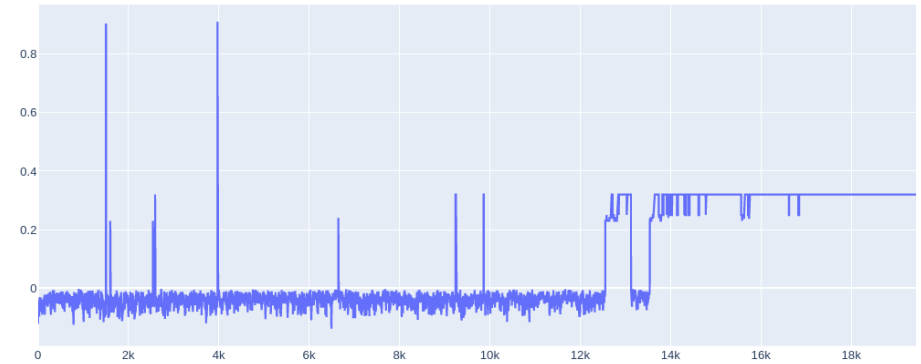



Problémom tohto prístupu boli veľké instancie. Na grafe nižšie je instancia s 75 premennými a 310 klauzulami. Algoritmus nebol schopný ani pri vysokej teplote nájsť žiadne riešenie.




Ďalším pokusom bolo aj prehľadávanie pomocou prevrátenia tej premennej, ktorá sa najčastejšie nachádza v opačnej polarite ako literál v klauzulách. To viedlo k uviaznutia v lokálnych extrémoch a predĺženiu algoritmu, pretože táto operácia bola oveľa náročnejšia na výpočet oproti výberu jednej náhodnej premennej a išlo to proti myšlienke častých malých rýchlych zmien. Tento postup sa neosvedčil.

Finálny postup bol riešením pre inštancie, ktoré mali veľké množstvo klauzulí aj premenných. Algoritmus nájde nesplnené klausule a uloží si premenné z týchto klauzulí. V každom kroku náhodne prevráti jednu z nich. Vďaka tomu sa algoritmus rýchlejšie dopracuje k splnenej klauzuli. Tento spôsob je nastavený ako predvolený. Za radu ďakujem mojim spolužiakom z FIT. Algoritmu stačia pomerne nízke teploty. Na grafe nižšie ukazujem situáciu, kde algoritmus našiel optimum už niekedy v prvej tretine výpočtu ale neskôr uviazol v inom lokálnom maxime. Možným riešením by bola detekcia intenzifikačnej fázy (veľmi nízke teploty) a následne vrátenie sa do predchádzajúceho lepšieho stavu a pokračovanie intenzifikácie v tejto oblasti.



**Chladenie**

Proces ochladzovania je znižovanie teploty podľa geometrického radu s koeficientom faktor ochladzovania ***k*** - pri každom ochladení sa aktuálna teplota vynásobí faktorom ochladzovania ***k***: $T_{i+1} = k * T_i$. V celom procese simulovaného ochladzovania si algoritmus uchováva najlepšie nájdené riešenie, ktoré potom vráti ako riešenie.

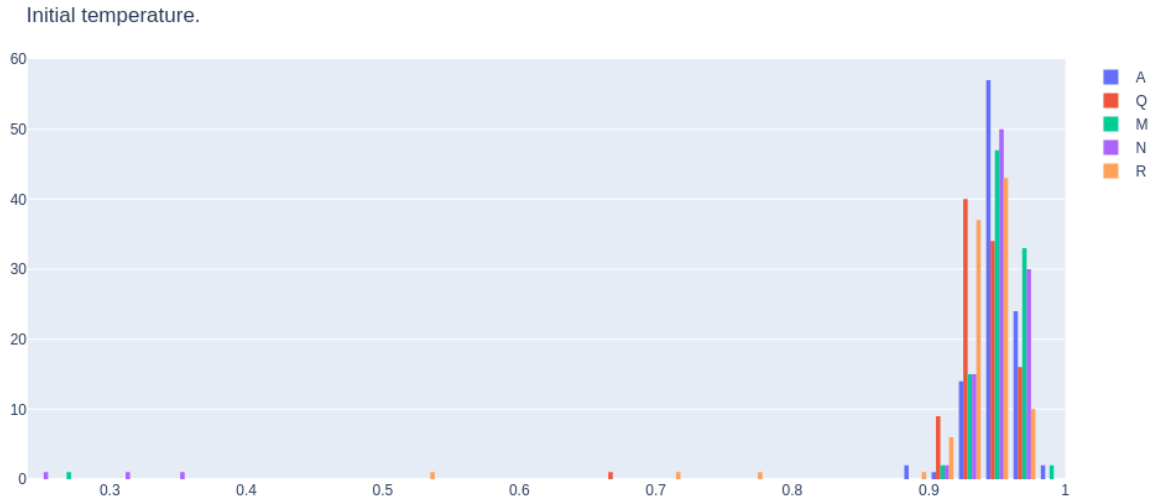
# Whitebox fáza

Počas whitebox fázy som používal 100 instancií z každého datasetu s 20 premenými a 78 resp. 88 klauzulami. V blackbox fáze som použil iných 100 instancií.

**Výpočet počiatočnej teploty**

Z počiatočného stavu sa vygenerujú všetky susedné stavy (je ich ***n*** podľa počtu premenných vo výroku). Každý z nich sa ohodnotí a vyberie sa najdrahší a najlacnejší stav, ich rozdiel bude ***0 < d < 1.*** Počiatočná teplota sa nastaví na ***1-d.*** Ak sú v okolí podobné stavy, ich rozdiel bude malý a teda počiatočná teplota bude vysoká. Ak je medzi susednými stavmi veľký rozdiel, počiatočná teplota bude nízka. Inšpirácia od Ben-Amaeur[1]

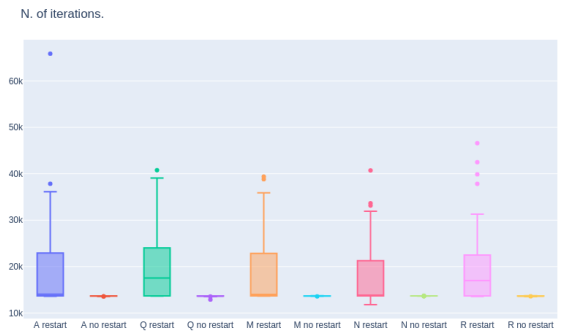
Podľa analýzy prvotnej teploty je vidno dôvod, prečo niektoré inštancie majú nižší počet iterácií. Väčšina instancií mala malý rozdiel medzi susednými stavmi a preto bola prvotná teplota medzi 0.9 a 1.0. Podľa náročnosti problému môžeme manuálne upravovať počet iterácií vo vnútornom cykle ***I*** alebo **α** (explorácia) alebo koeficient chladenia ***k*** (intenzifikácia).

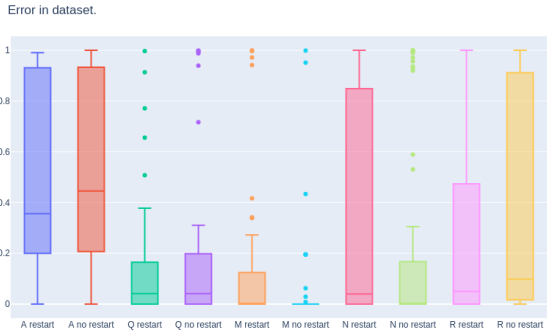



[1] Ben-Ameur, W. Computing the Initial Temperature of Simulated Annealing. *Computational Optimization and Applications* **29,** 369–385 (2004). https://doi.org/10.1023/B:COAP.0000044187.23143.bd

**Reštart**

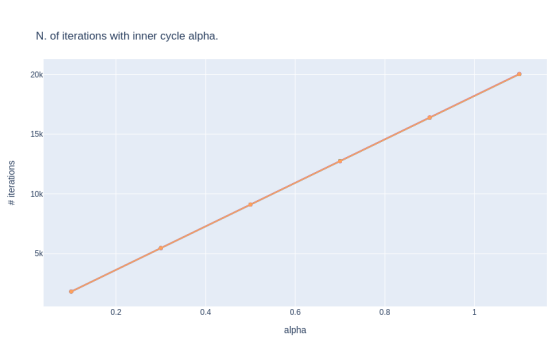
Pri prehľadávaní veľkého stavového priestoru sa môže riešenie nachádzať ďaleko od počiatočného stavu. Algoritmus bude znižovaním teploty intenzifikovať oblasť, ktorá neobsahuje riešenie. Reštart umožní znovu hľadať riešenie z iného počiatočného stavu. Ak algoritmus dlho nevie nájsť lepšie riešenie (#zamietnutých stavov) a zároveň stav, v ktorom sa nachádza, nespĺňa výrok, je malá pravdepodobnosť (1 ku 1000), že nastane reštart aj s novým nastavením teploty. Toto riešenie môže spôsobiť veľmi dlhé výpočty, z toho dôvodu bol zhora obmedzený počet iterácií vysokou hodnotou (100 000 pre 20 premenných). Ak algoritmus dokáže nájsť vo väčšine prípadov riešenie, toto spomalenie by sa mohlo stratiť. Ak nie, je odôvodniteľné, že problém je veľmi ťažký a algoritmus potrebuje viac času na výpočet. Reštarty sa neukázali ako vhodné. Reštarty sú dôvodom pre veľký nárast počtu iterácií ale v priemere neznížili markantne chybu výpočtu ako vidno na grafe nižšie. Algoritmus má prednastavené zakázané reštarty.

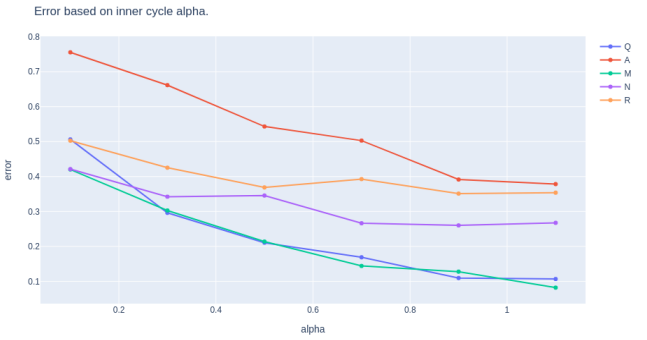



**Equilibrium α**

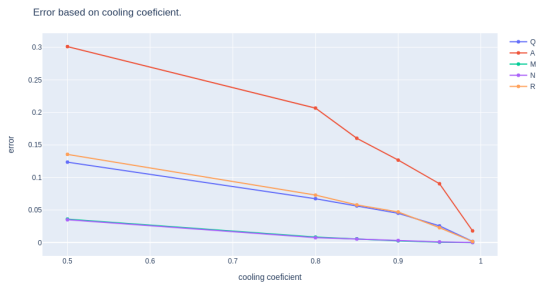
Určí počet navštívených stavov (iterácií) vo vnútornej slučke pri konštantnej teplote - slúži na exploráciu. Počet iterácií $I$ závisí od počtu premenných $n$ vo výroku a počtu klauzúl $c$, kde počet iterácií $I = c*n*\alpha$. Na grafe je vidno, že vyšší počet iterácií vo vnútornom cykle lineárne zvyšuje počet iterácií a zvoľna znižuje chybu. Prednastavená hodnota je 0.9, je možnosť manuálne zvoliť počet iterácií $I$ alebo určiť hodnotu $\alpha$. Testované boli hodnoty 0.1, 0.3, 0.5, 0.7, 0.9, 1.1. V tomto testovaní bola použitá prvotná stratégia (prevrátenie náhodnej jednej premennej) na prehľadávanie priestoru. V ďalších testoch je už použitá ako štandardná stratégia hľadanie nesplnených klauzulí a prevrátenie premennej, ktorá ich splní.

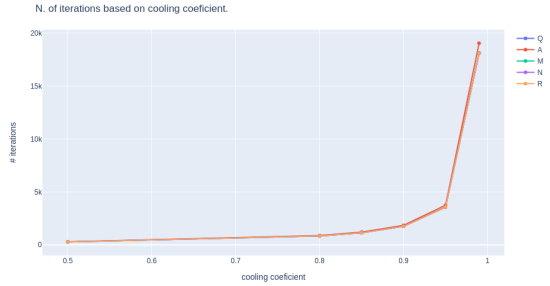



### Faktor ochladzovania *k*

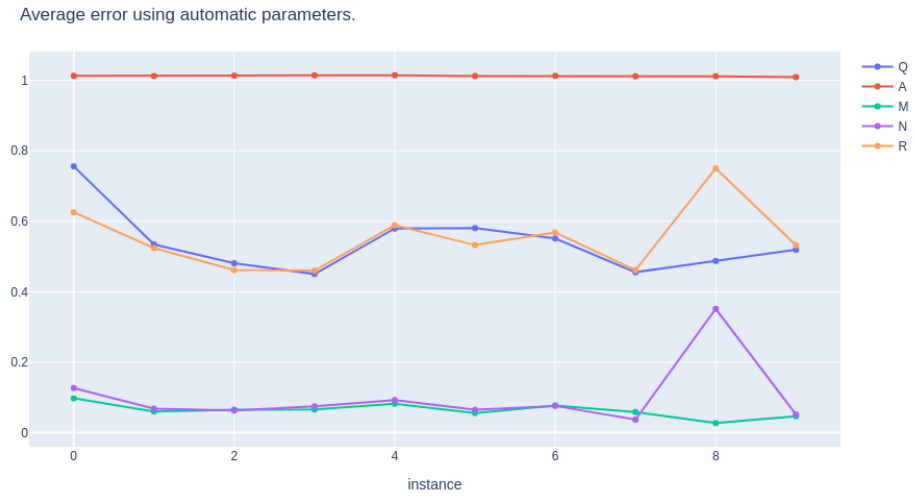
Určuje proces chladenia, po opustení slučky equilibria zníži teplotu koeficientom ***k***. Má exponenciálny vplyv na počet iterácií algoritmu. Prednastavenú hodnotu som určil empiricky pomocou testovania na 0.99. Testované boli hodnoty 0.5, 0.8, 0.85, 0.9, 0.95, 0.99.



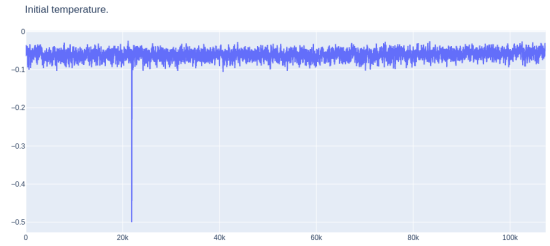




### Problém s veľkými instanciami

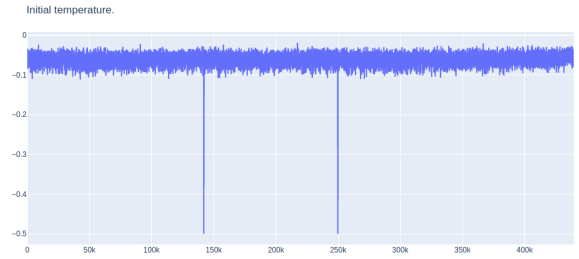
V datasete s veľkým počtom premenných (100 pre A) a klauzúl (400 pre A) som objavil, že algoritmus nenachádza riešenia. V popise grafu nižšie je chyba, nejedná sa o *error* ale o hodnotu ***( 1 - cena stavu).*** Ak stav nespĺňa výrok, má zápornú hodnotu a výsledná hodnota je vyššia ako 1. Vysoké hodnoty (ale menšie ako 1) preto znamenajú nízku cenu nájdeného riešenia, nízke hodnoty naopak vysokú. Tento výpočet bol dlhý a preto som neopakoval vytváranie grafu. Každú instanciu som vypočítal 3x a spriemeroval.



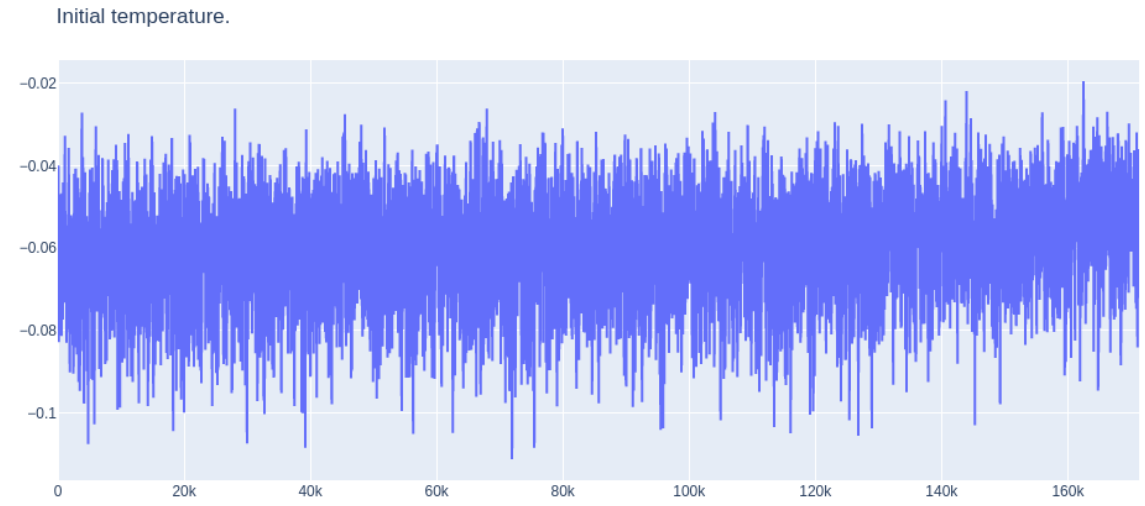
Možným riešením by bolo použitie reštartov, ktoré sa mi neosvedčilo. Ukážkový beh, kde algoritmus 1x reštartoval (-0.5 na na pozícii 20k).




Skúšal som násilne začínať s vysokou teplotou, reštartovať, prehľadávať 2x väčšie okolie ($\alpha = 2$), ale neprinieslo to výsledky.




Rovnako zvýšiť intenzifikáciu nepomohlo, nastavenie ***k=0.995.***




Môj návrh by bol použiť tzv. Unit propagation. Skúšal by som nastaviť jednu z trochu premenných v klauzuli a sledoval jej výskyt v ďalších klauzuliach. Ak by nastala situácia, že niektorá klauzula závisí od mojej premennej, musel by som jej nastaviť fixnú hodnotu a dokázal by som orezať priestor. Takisto nastavením tejto premennej by som sledoval výskyt a závislosti v ďalších klauzuliach atď. Ide o dobre preskúmaný prístup, hlavne CDCL riešičoch. Toto riešenie som počul na predmete umelej inteligencie (NI-UMI), spolu s ďalšími local-search metódami.

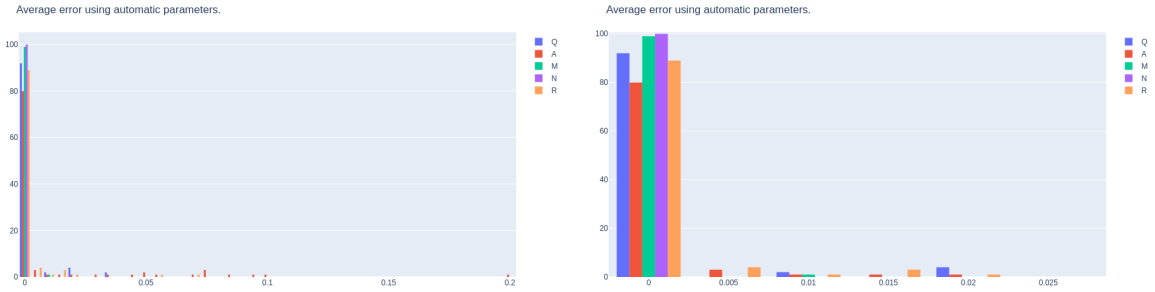
# Blackbox fáza

**Popis datasetu**

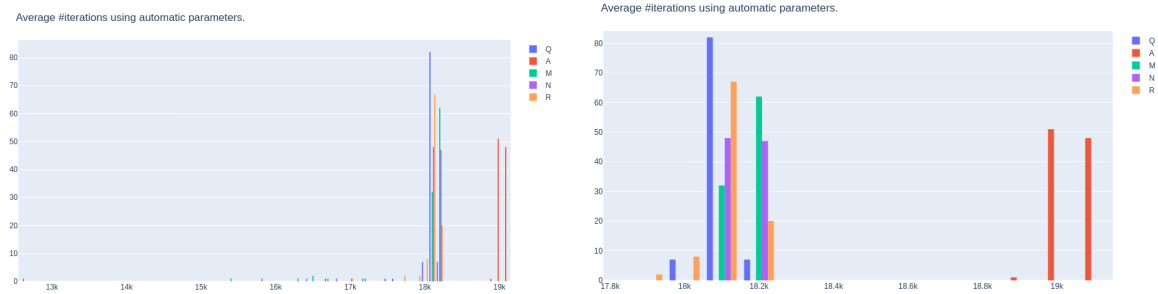
Testovanie prebiehalo na instanciách z datasetov *A, Q, M, N, R*, s **20** premennými a **78** (pre A 88) klauzulami v instancii.

| Názov datasetu (100 inst.) | Avg. relative error | Avg. num. of iterations |
|---|---|---|
| A | 0.0110 | 19045 |
| Q | 0.0017 | 18074 |
| M | 0.00007 | 18060 |
| N | 0.0 | 18030 |
| R | 0.002 | 18086 |

Na grafe nižšie je vidno, že najvyššie hodnoty relatívnej chyby boli 0.2 a väčšina z nich bola okolo nuly. Vpravo je detail na histogram relatívnej chybovosti k optimálnym výsledkom.




Histogram počtu iterácií podľa datasetov, vpravo detail. Je vidno, že dataset A (červená) potreboval na výpočet najviac iterácií.



Graf relatívnej chyby na inštanciách zobrazuje, že najväčší rozptyl v chybe má dataset *A*. Ostatné datasety sa iba veľmi zriedka dostali nad 5% chybu.

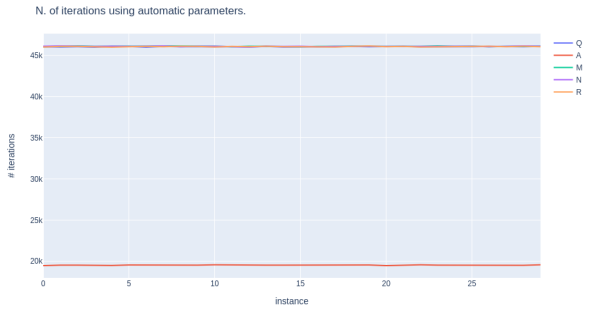

## Škálovanie

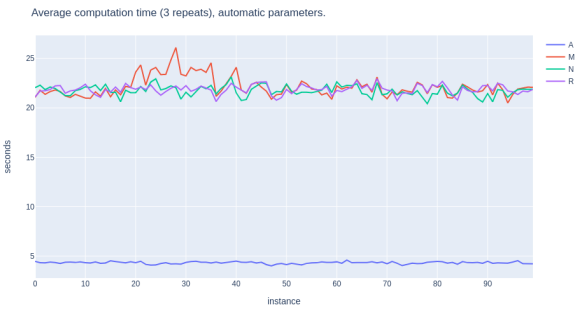
Testovanie škálovania prebiehalo na instanciách z datasetov *A*, *Q*, *M*, *N*, *R*, s **50** (20 pre A) premennými a **201** (pre A 91) klauzulami v instancii.

| Názov datasetu | Avg. relative error (30 inst.) | Avg. num. of iterations (30 inst.) | Avg. computation time in seconds (100 inst.) |
|---|---|---|---|
| A (20p, 91k) | 0.02 | 19500 | 4.8 |
| Q | 0.09 | 46000 | 23.3 |
| M | 0.0019 | 46100 | 22.9 |
| N | 0.004 | 46100 | 22.9 |
| R | 0.09 | 46000 | 23.3 |


Chyba výpočtu je vyššia, miestami aj o jeden rád. Myslím si, že je to spôsobené nedostatočným škálovaním môjho algoritmu ku prehľadávaniu okolia - vždy prevraciam iba jednu hodnotu. Ak by som v každom kroku prevrátil napríklad 1/10 premenných, algoritmus by mohol priniesť lepšie výsledky.

Prehľadávané okolie sa zväčšuje vzhľadom ku počtu premenných a klauzulí ($n*c*\alpha$), preto si myslím, že teplotu ani faktor ochladzovania netreba meniť s veľkosťou instancie.




## Diskusia

Algoritmus vykazuje nárast chyby pri zväčšení instancií. Myslím si, že sa jedná o systematickú chybu v prehľadávaní okolia - algoritmus v každom kroku prevráti náhodne presne jednu premennú. Pri instanciách s 20 premennými sa tak jedná o 5% všetkých premenných, čo je pomerne veľká zmena. Naopak, pri instanciách s 90 premennými by táto zmena nemusela dostatočne pokryť stavový priestor. Možným riešením by bolo dynamicky nastavovať množstvo prevrátených premenných alebo hľadanie spôsobu ako splniť nesplnené klauzule inteligentnejšie (unit propagation).

# Záver

V tomto algoritme som použil heuristickú metódu simulovaného ochladzovania na hľadanie riešenia váženého 3-SAT problému s automatickým nastavovaním parametrov. Výsledky ukazujú úspešné nasadenie pre problém s 20 a 50 premennými ale pre problémy so 75 premennými sa objavili vážne problémy - algoritmus nenašiel žiadne riešenie.
Čiastkové pomocné metódy použité v tomto algoritme boli:

A. automaticky nastavená počiatočná teplota na základe rozdielu ceny stavov v počiatočnom riešení,
B. Nelineárna cenová funkcia, ktorá je normovaná do [-1, 1] a umožňuje porovnávať aj stavy, ktoré nespĺňajú výrok,
C. Implementované reštarty a detekcia ich nasadenia v prípade dlhodobého neúspešného hľadania výsledku,
D. Dynamické nastavenie prehľadávanie okolia na základe počtu premenných a klauzulí, s manuálnym škálovaním a empiriciky zvolenou hodnotou,
E. Stratégia prehľadávania okolia, ktorá rozlišuje stavy, kde je nájdené riešenie a kde nie je. V prvom prípade sa snaží náhodne prevrátiť jednu hodnotu a prehľadávať priestor v okolí. V druhom prípade, keď nie je výrok splnený, sa sústredí na premenné v nesplnených klauzuliach a tým pádom sa snaží výrok najprv splniť a nájsť riešenie.