Umelá inteligencia: Zadanie I.

Problém obchodného cestujúceho – Genetický Algoritmus & Simulované Žíhanie

Ondrej Krajčovič

Zadanie:

Je daných aspoň 20 miest (20 – 40) a každé má určené súradnice ako celé čísla X a Y. Tieto súradnice sú náhodne vygenerované. (Rozmer mapy môže byť napríklad 200 * 200 km.) Cena cesty medzi dvoma mestami zodpovedá Euklidovej vzdialenosti – vypočíta sa pomocou Pytagorovej vety. Celková dĺžka trasy je daná nejakou permutáciou (poradím) miest. Cieľom je nájsť takú permutáciu (poradie), ktorá bude mať celkovú vzdialenosť čo najmenšiu.

Výstupom je poradie miest a dĺžka zodpovedajúcej cesty.

Ondrej Krajčovič

I. Všeobecné informácie:

Moje riešenie pozostáva z štyroch python súboroch typu *.py*:

- *main.py*
- *GLOBAL_VARIABLES.py*
- *geneticAlgorithmFunctions.py*
- *simulatedAnnealingFunctions.py*

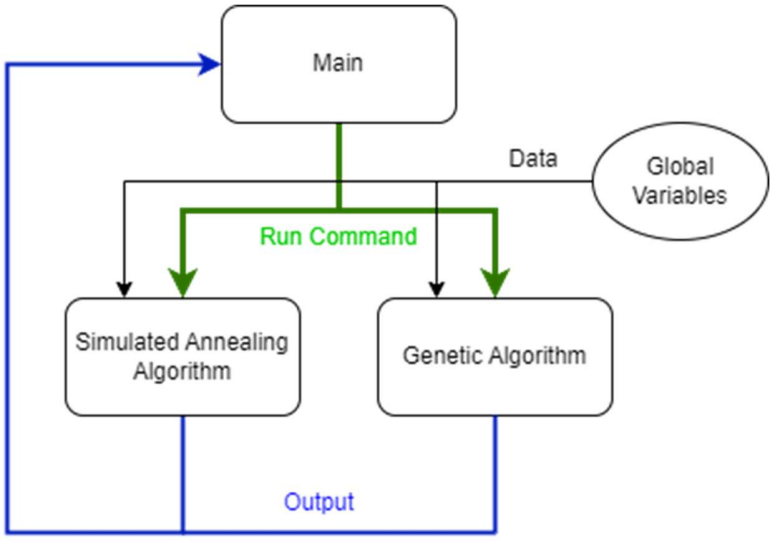
Úlohou programu *main.py* je spustiť oba algoritmy ako funkcie, zmerať čas za ktorý prebehnú a na výstupe ho zobraziť spolu s najlepším riešením

V *GLOBAL_VARIABLES.py* sa nachádza definícia dôležitých globálnych premenných, ktoré určujú vlastnosti algoritmov, ako aj generáciu náhodného grafu

V *geneticAlgorithmFunctions.py* a *simulatedAnnealingFunctions.py* sa nachádzajú jednotivé algoritmy, ich pomocné funkcie a lokálne premenné, ako aj zakomentované pomocné podprogramy, určené na zobrazenie priebehu funkcií a podrobnejšieho výstupu

Do *main.py* sú na začiatku vložené funkcie z *geneticAlgorithmFunctions.py* a *simulatedAnnealingFunctions.py*. Okrem nich používam rôzne python knižnice a to konkrétne:

- *random*
- *time*
- *math (sqrt() – druhá odmocnina, exp() – exponent)*
- *tkinter*



Obr.1: všeobecná schéma riešenia

II. Genetický Algoritmus(Genetic Algorithm):

Opis Algoritmu: Genetický Algoritmus (ďalej GA, alebo Genetic Algorithm) je založený na Darwinovej teórií o evolúcií a spadá medzi Evolučné algoritmy. Pracuje s populáciou potenciálnych riešení, spomedzi ktorých selektuje najlepších jedincov, kríži ich (kombinuje ich vlastnosti), mutuje (umelo vlastnosti mení), a následne porovnáva medzigeneračný pokrok bodovaním (ďalej fitness) až pokým nenájde relatívne optimálne riešenie

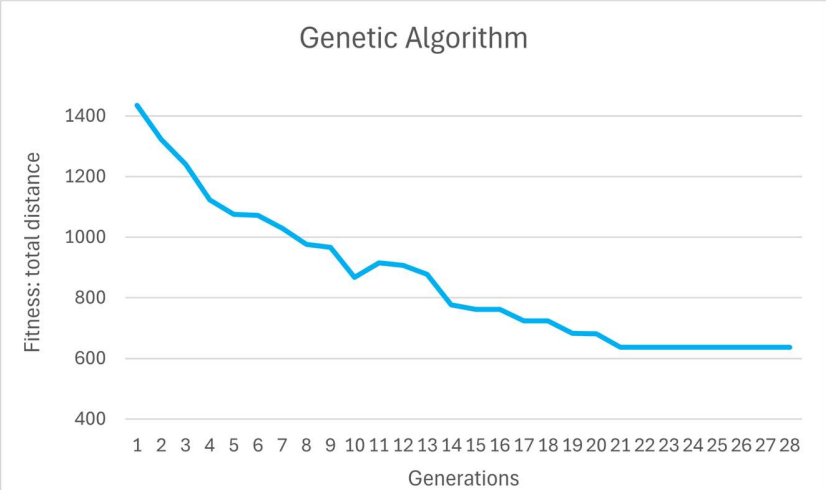
Špecifiká Genetického Algoritmu: každá generácia obsahuje 512 jedincov, celkový počet generácií je maximálne 1000.

Ak však má sedem generácií po sebe rovnaký výsledok, program končí a vracia najlepšiu cestu.

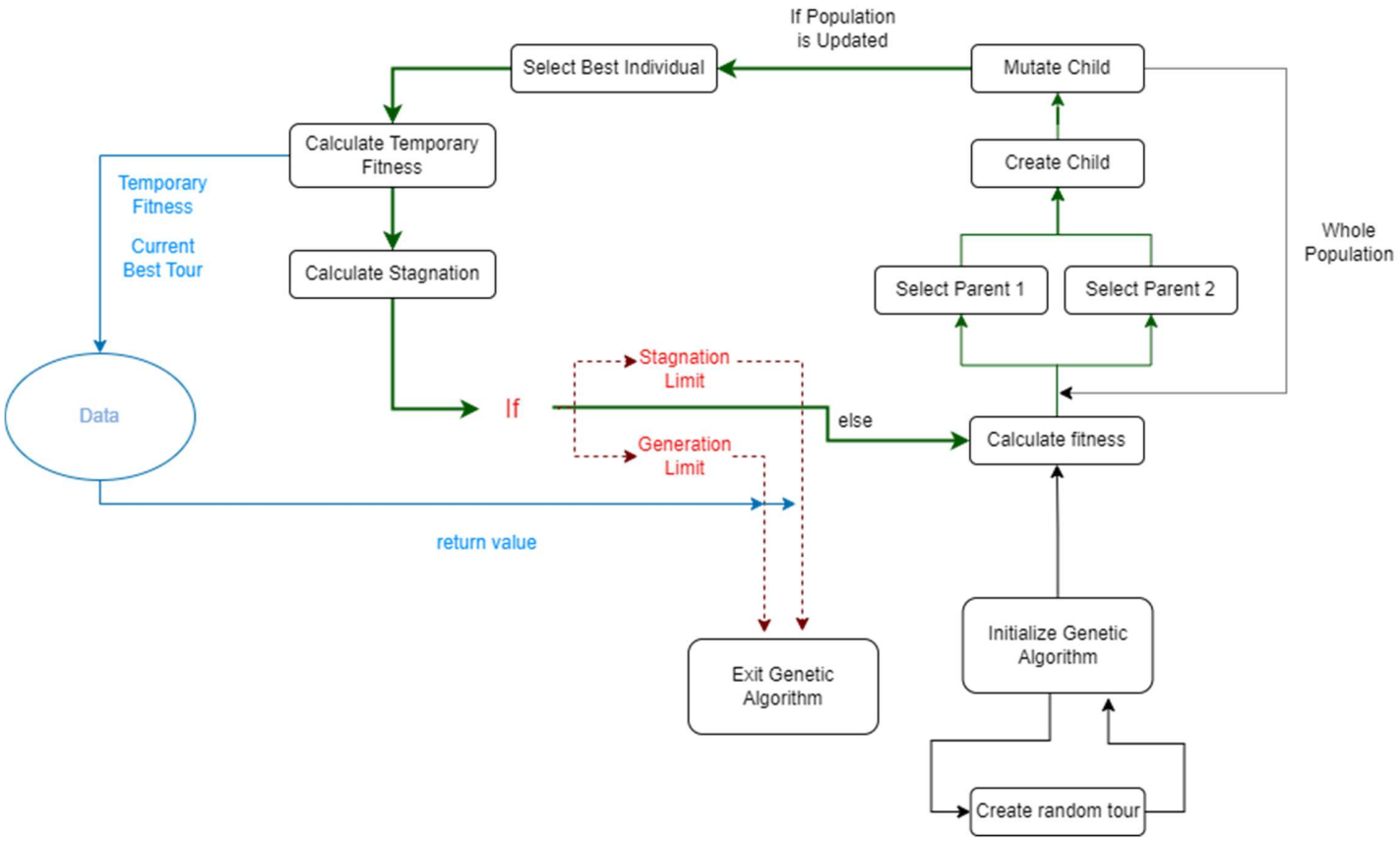
Pri výbere na kríženie používam Turnajový výber (náhodne sa vyberú viaceré riešenia a najlepšie z nich pokračuje ku kríženiu).

Kríženie je uskutočnené tým, že z prvého rodiča vyberieme náhodný interval, ten vložíme dieťaťu, pričom zvyšok trasy dieťaťu dáme od druhého rodiča.

Pri mutácii dochádza k náhodnej výmene hrán s pravdepodobnosťou mutácie 0,1 %. Fitness je určená celkovou euklidovskou dĺžkou obvodu trasy, kalkulujem ju vo vlastnej podfunkcií



Obr.2: Graf Fitness jednolivých generácií v Genetickom Algoritme



Obr.3: schéma môjho riešenia pre Genetický Algoritmus

III. Simulované Žíhanie (Simulated Annealing):

Opis algoritmu: Simulované žíhanie (ďalej napr.: Simulated Annealing) je algoritmus inšpirovaný žíhaním,patrí medzi Riadené náhodné vyhľadávacie techniky (Guided random search techniqes).

Žíhanie je proces postupného ochladzovania kovov za účelom dosiahnutia lepšej formovateľnosti a menšej tvrdosti kovov. postupným znižovaním teploty sa pohyb častíc na mikroskopickej úrovni redukuje.

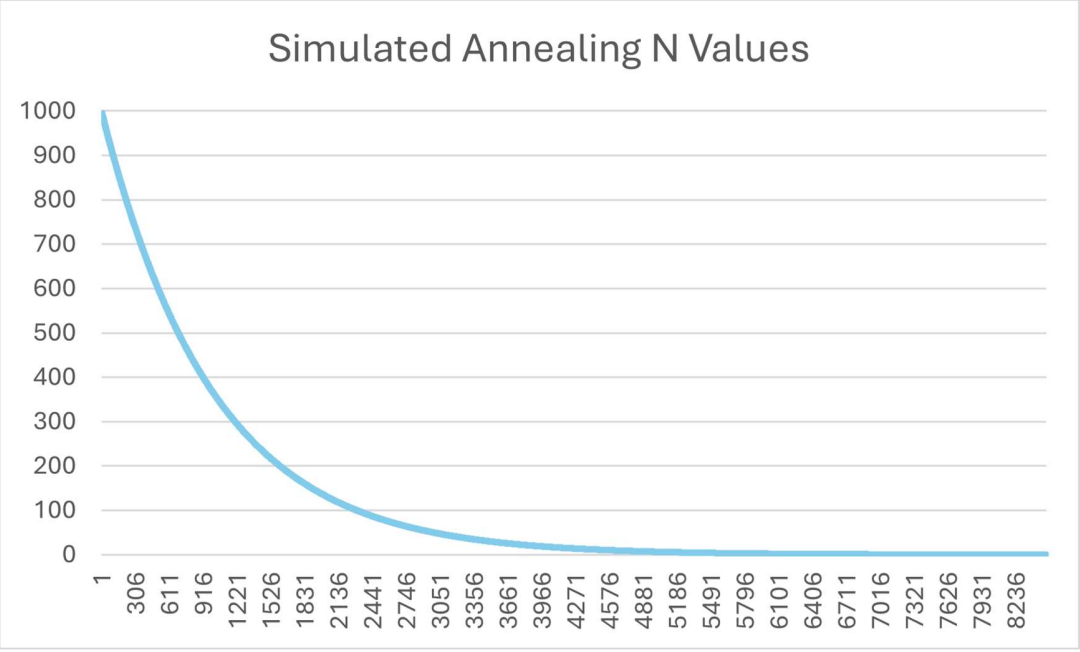
Algoritmus využíva postupné znižovanie premennej teploty, pričom teplota určuje pravdepodobnosť radikálnej zmeny trasy. Na začiatku je teda program náchylný robiť väčšie zmeny, neskôr sa trasa relatívne ustáli na optimálnom riešení

Špecifiká implementácie:

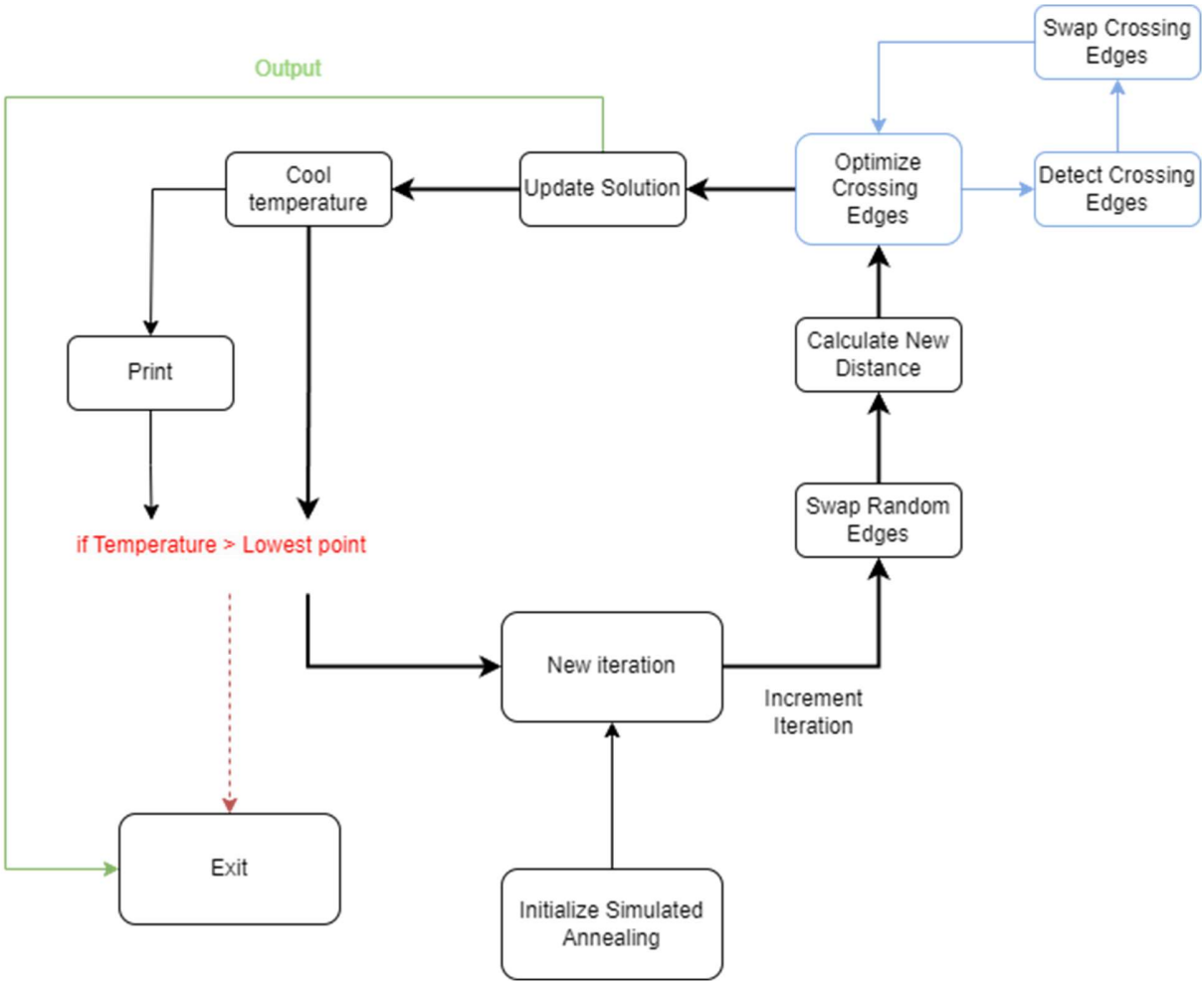
Počiatočná teplota je staticky inicializovaná na 1000, pričom každou iteráciou sa násobí konštantou 0,999.Teplota sa znižuje až pokým nie je menšia ako 0,2 – prebehne teda ~ 8000 iteriácií.

Po generovaní zmien v grafe na základe teploty sa v mojej implementácií uskutoční optimalizácia riešenia. Pomocná funkcia prechádza grafom a detekuje krížiace sa hrany pomocou vektorového zobrazenia v kombinácií so súradnicami bodov. Ak funkcia nájde krížiace sa hrany, vymení ich tak aby sa nekrížili.Táto optimalizácia výrazne prispela k efektivite implementácie Simulovaného Žíhania.

Následne program redukuje teplotu, a pokračuje do ďalšej iterácie



Obr.4: Graf ochladzovania premennej N postupom času



Obr.5: schéma môjho riešenia pre Simulované Žíhanie

IV. Záver:

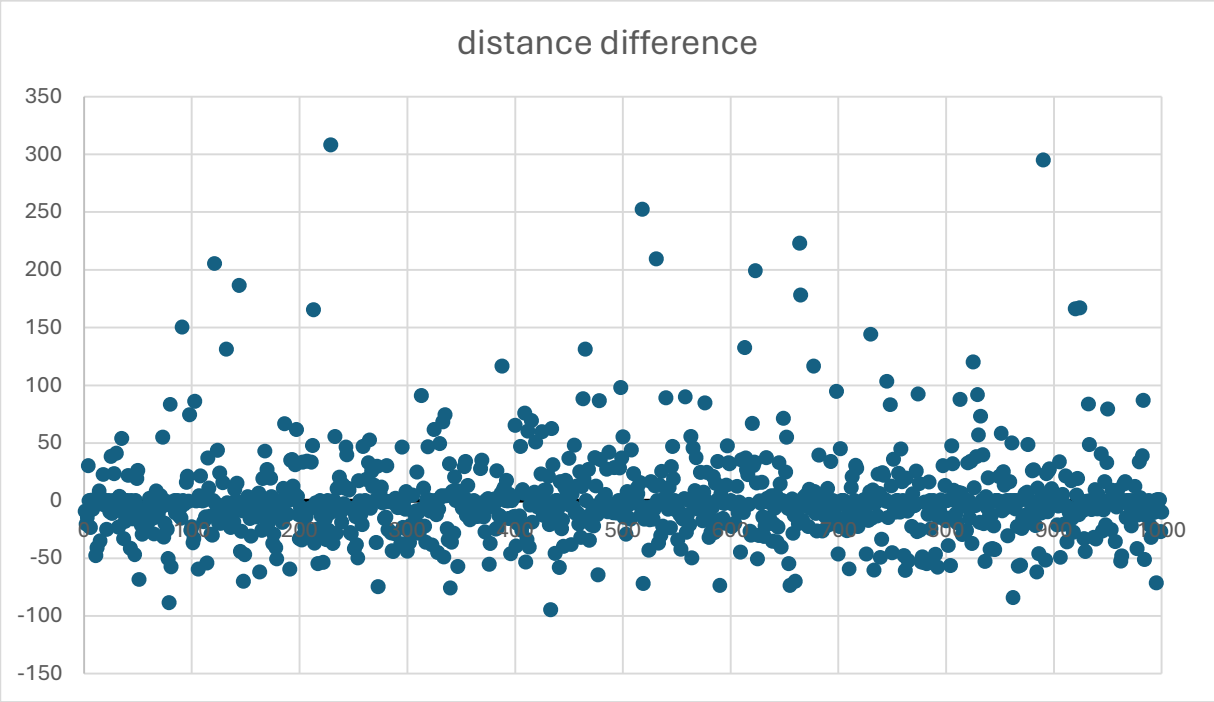
Pomocným programom som 1000-krát spustil main.py, to jest oba algoritmy a porovnával som dĺžku cesty nájdenej jednotlivými algoritmami a čas za ktorý danú cestu našli.
Výsledkom bolo nasledovné:

Priemerný čas behu programu bol pri Simulovanom Žíhaní o približne 0.31 s kratší pri priemernej dobe ~0.61 s, to jest skoro o polovicu rýchlejšie. ***Simulované Žíhanie je teda časovo efektívnejšie***

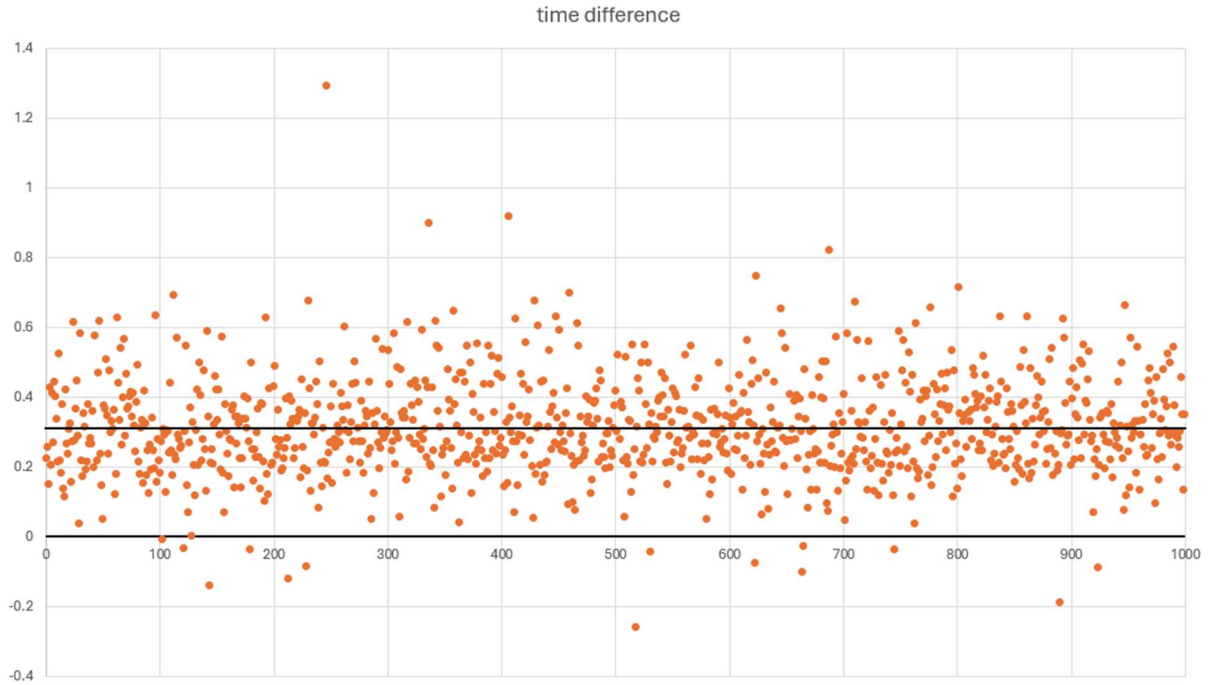
Priemerná dĺžka trasy bola približne rovnaká, mierne lepšia pri simulovanom žíhaní (o ~0.71 ku ~744 priemernej dĺžky trasy). ***Správnosť výsledku bola približne rovnaká***

GA čas	GA dĺžka	SA čas	SA dĺžka	Časový rozdiel	Rozdiel dĺžok
0.762734	744.9841	0.45213	744.2697	0.310604	0.714421
				SA	SA

Tabuľka 1:Priemerné namerané hodnoty vlastností algoritmov po 1000 spusteniach



Obr.6: Graf nameraných hodnôt rozdielov dĺžky trás Simulovaného Žíhania a Genetického Algoritmu, kedy x-os určuje množstvo spustení (0 - 1000), y-os určuje rozdiel nameraných hodnôt(-150 - 350)



Obr.6: Graf nameraných hodnôt rozdielov času pribehu Simulovaného Žíhania a Genetického Algoritmu, x-os určuje množstvo spustení (0 - 1000), y-os určuje rozdiel nameraných hodnôt(-0,4s - 1,4s)