Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program : Aplikovaná informatika

Obor: Informační systémy a technologie

Metoda nejmenších čtverců genetickým algoritmem

DIPLOMOVÁ PRÁCE

Student : Bc. Matúš Holec

Vedoucí : RNDr. Vladimír Tichý

Oponent : Ing. Tomáš Šalamon, MBA, Ph.D.

2012

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 25.4.2012 . . . . . .

Matúš Holec

Poděkování

Rád by som poďakoval svojmu vedúcemu, RNDr. Vladimírovi Tichému, za hodnotné pripomienky a rady, ktoré mi poskytoval počas priebehu tvorby práce.

Abstrakt

Táto diplomová práca sa zaoberá návrhom a implementáciou genetického algoritmu na aproximáciu nelineárnych matematických funkcií použitím metódy najmenších štvorcov. Jedným z cieľov tejto práce je teoreticky popísať základy genetických algoritmov. Druhým cieľom je vytvorenie programu, ktorý by bol potenciálne využívaný vedeckými ústavmi na aproximáciu empiricky nameraných dát. Textová časť práce sa okrem teoretického popisu problematiky zaoberá hlavne návrhom genetického algoritmu a celej aplikácie riešiacej daný problém. Špecifikom zadania je nutnosť aproximácie hodnôt rôznymi matematickými funkciami na viacerých intervaloch, a následná spojitosť týchto funkcií. Takúto funkcionalitu neponúka žiadny dostupný softvér.

Klíčová slova

genetický algoritmus, metóda najmenších štvorcov, nelineárna aproximácia, fitness, populácia

Abstract

This thesis describes the design and implementation of genetic algorithm for approximation of non-linear mathematical functions using the least squares method. One objective of this work is to theoretically describe the basics of genetic algorithms. The second objective is to create a program that would be potentially used to approximate empirically measured data by the scientific institutions. Besides the theoretical description of the given subject, the text part of the work mainly deals with the design of the genetic algorithm and the whole application solving the given problem. Specific part of the assignment is that the developed application has to support approximation of points by various mathematical non-linear functions in several different intervals, and then it has to insure, that resulting functions are continuous throughout all the intervals. Described functionality is not offered by any available software.

Keywords

genetic algorithm, least squares method, non-linear approximation, fitness, population

Obsah

1 Úvod 6
1.1 Téma práce 6
1.2 Ciele práce 7
1.3 Predpoklady a obmedzenia práce 8
1.4 Výstupy a očakávané prínosy práce 8

2 Práce s podobnou tematikou 10
2.1 Praktické aplikácie genetických algoritmov 10
2.1.1 Aproximácia empiricky získaných dát 10
2.1.2 Optimalizácia sietí 11
2.1.3 Tvorba rozvrhov 11
2.1.4 Inžinierske aplikácie 12
2.1.5 Definícia pravidiel 13
2.1.6 Predpoveď chovania cien akcií 13
2.1.7 Vyvíjajúci sa hardware 14
2.1.8 Ďalšie aplikácie genetických algoritmov 14
2.2 Praktické aplikácie metódy najmenších štvorcov 15

3 Genetické algoritmy 16
3.1 Zaradenie genetických algoritmov 17
3.2 Základné pojmy 19
3.3 Štruktúra genetického algoritmu 21
3.3.1 Reprezentácia jedincov 22
3.3.2 Ohodnotenie jedincov 23
3.3.3 Selekcia rodičov 24
3.3.4 Rekombinácia 26
3.3.5 Mutácia 28
3.3.6 Selekcia jedincov pre novú generáciu 29
3.4 Postup návrhu genetického algoritmu 30

4 Metóda najmenších štvorcov 31
4.1 Metóda najmenších vzdialeností 32

5 Požiadavky na aplikáciu 34
5.1 Nefunkčné požiadavky na systém 34
5.2 Funkčné požiadavky na systém 35
5.3 Prípady užitia systému 37

6 Analýza a návrh riešenia 40
6.1 Analýza a návrh technológie 40

6.2 Analýza a návrh grafického rozhrania....43
6.3 Logický návrh....46
6.3.1 Hlavné okno....47
6.3.2 Konfigurátor....47
6.3.3 Plotter grafu....49
6.3.4 Aproximátor....49
6.3.5 Matematické funkcie....49
6.4 Návrh genetického algoritmu....50
6.4.1 Voľba reprezentácie problému....51
6.4.2 Pôvodná inicializácia populácie....52
6.4.3 Spôsob ohodnocovania jedincov populácie....53
6.4.4 Selekcia párov rodičovskej populácie....55
6.4.5 Kríženie rodičovských párov....56
6.4.6 Mutácia....57
6.4.7 Spôsob obnovy populácie....57
6.4.8 Ukončenie algoritmu....58
6.4.9 Spojitosť aproximovaných funkcií na intervaloch....58

7 Vývoj programu....60
7.1 Programovací jazyk a vývojové prostredie....60
7.1.1 Framework Qt....61
7.2 Použité technológie....62
7.2.1 Parsovanie konfiguračného súboru....62
7.2.2 Plotter grafu....63
7.2.3 Genetický algoritmus....64
7.3 Implementácia....65
7.4 Parametre implementovaného genetického algoritmu....68
7.5 Ukážka programu....71

8 Záver....74

Terminologický slovník....75

Zoznam literatúry....76

Zoznam obrázkov....79

1 Úvod

Názorov na to, ako sa dokázali živé tvory tak vhodne adaptovať na svoje prostredie, ako aj prečo tie, ktoré sa neprispôsobili, vyhynuli, bolo počas histórie ľudstva mnoho. Ako najvhodnejšou a zároveň najlogickejšou sa ukázala byť Darwinova evolučná teória. Princíp prirodzeného výberu v prírode a prežitia najprispôsobivejšieho jedinca bol formulovaný Charlesom Darwinom v roku 1859, dávno predtým ako boli známe mechanizmy genetiky. Spočiatku bolo na tento princíp nazerané skepticky, avšak postupným objavovaním nových zákonitostí dedičnosti a genetiky nebolo o korektnosti Darwinových záverov pochýb.

Evolučná teória sa v posledných desaťročiach začala využívať aj pri riešení problémov pomocou informačných technológií, ktorých riešenie bežnými analytickými, prípadne numerickými metódami zabralo neúnosne veľa času, alebo vôbec neexistovalo. Princípom prežitia toho jedinca, ktorý sa dokáže najlepšie adaptovať na okolité podmienky sa inšpirovala oblasť evolučných algoritmov. Ich základnou charakteristikou je fakt, že systémy, ktoré ich využívajú, uchovávajú populáciu potenciálnych riešení, medzi ktorými dochádza ku iteratívnym evolučným procesom kríženia, mutácie, dedičnosti, obmeny generácií a selekcie na základe najlepšej hodnoty predpokladu prežitia jedinca, jeho ohodnoteniu – tzv. fitness hodnoty. Dochádza tu ku súťaženiu medi jedincami v populácií, kedy podobne ako v prírode, sa do procesu tvorby nasledujúcej, silnejšej generácie riešenia zapojujú najvhodnejší jedinci, pričom ich selekcia spočíva práve na schopnosti poskytnúť čo najviac optimálne riešenie zadania. Takýmto spôsobom sa postupne systémy dopracujú ku najvhodnejšiemu riešeniu. Špecifickú, ale zároveň veľmi obsiahlu časť evolučných algoritmov tvoria genetické algoritmy. Okrem evolučných procesov kríženia, mutácie, selekcie a obmeny generácií tu navyše zohrávajú úlohy gény a genómy, ktoré reprezentujú riešenia úlohy.

1.1 Téma práce

Hlavnou témou práce je oblasť genetických algoritmov. Tematicky sú tieto algoritmy súčasťou metód umelej inteligencie. Táto oblasť je však natoľko široká, že v práci sa budem prakticky zaoberať možnosťou aproximácie nelineárnych matematických funkcií pomocou metódy najmenších štvorcov s využitím genetických algoritmov. Do tejto oblasti spadá aj preskúmanie aplikácie aproximačnej metódy najmenších štvorcov na problémy riešené pomocou genetických algoritmov. Zadanie práce bolo dodané Akadémiou Vied Českej Republiky v Brne. Jedná sa o vytvorenie programu na aproximáciu empiricky nameraných dát. Zadanie je bližšie špecifikované v kapitole 5. Danú úlohu nie je možné, z dôvodu vyššieho počtu aproximovaných matematických funkcií a ich následnej nutnosti spojitosti na zadaných intervaloch, implementovať pomocou bežných numerických, prípadne analytic-

kých metód. Z toho dôvodu sa použitie genetických algoritmov na riešenie tohto problému javí ako najvhodnejší prístup.

Genetické algoritmy sa používajú hlavne na riešenie optimalizačných úloh, medzi ktorými môžu byť problémy typu tvorby rozvrhov, návrh kabeláže sietí, adaptívne riadenie, kognitívne modelovanie, prípadne optimalizácia databázových dotazov [1]. Genetické algoritmy na druhej strane nie je vhodné brať za plnohodnotný optimalizačný nástroj. Majú čiastočne stochastický charakter, a z toho dôvodu nie je možné garantovať, že ponúkané riešenie je optimálne. Element náhodnosti má vplyv aj na nejednotnosť riešenia v prípade viacerých behov systému využívajúceho genetické algoritmy, kde sa jednotlivé riešenia približujú optimu. Najvhodnejšie sa preto ukazuje hovoriť o týchto algoritmoch ako o simulátoroch prírodných procesov, schopných dopracovať sa k optimálnemu riešeniu.[2] Navzdory tomu bola v [3] dokázaná ich vysoká spoľahlivosť v nachádzaní optimálnych riešení. Aj zadanie práce má optimalizačný charakter, kde je potrebné odhadnúť optimálne parametre aproximovanej nelineárnej matematickej funkcie.

Téma práce obsahuje dva na seba logicky nadväzujúce väčšie celky. Rozdelenie týchto celkov zároveň predstavuje štruktúru práce. Prvým je teoretický popis riešenej problematiky. V rámci tohto popisu sú čitateľovi predstavené praktické problémy, ktoré už boli v tejto oblasti riešené. Ďalej sa prvý celok skladá z teoretického popisu, ktorý predstavuje základné logické stavebné celky genetických algoritmov a následne ich dáva do širších súvislostí, aby bolo možné pochopiť aplikácie týchto algoritmov. Ďalšou časťou teoretického celku je popis matematickej aproximácie funkcií s dôrazom na objasnenie princípov metódy najmenších štvorcov. Na východiská popísané v prvej časti nadväzuje druhý celok, ktorý sa venuje popisu postupu riešenia problému zadaného fyzikálnym ústavom AVČR v Brne. Tento celok začína popisom zadania od AVČR a zároveň všeobecným prehľadom fyzikálnej podstaty riešeného problému, vďaka ktorému čitateľ lepšie porozumie významu a originalite vytvorenej práce. Je tu predstavený postup pri analýze problému, návrhu riešenia a jeho implementácii a testovania programu. V závere práce sú zhrnuté dosiahnuté výsledky, skutočné prínosy práce, jej využitie ako aj možnosti rozšírenia.

Danú tému som si zvolil na základe môjho záujmu o vedeckú aplikáciu poznatkov z počítačovej vedy. Genetické algoritmy sa javia byť ako veľmi silný nástroj na riešenie širokej palety úloh, preto mi prišlo vhodné sa s týmto fenoménom bližšie oboznámiť a ich aplikáciu na matematickú problematiku si odskúšať. Zároveň som hľadal tému, ktorá by mala určitý vecný prínos a zároveň by mi pomohla rozšíriť si znalosti z oblasti programovania a tvorby aplikácií s užívateľským rozhraním.

1.2 Ciele práce

Prvým cieľom práce je dostatočne teoreticky popísať matematickú podstatu metódy najmenších štvorcov a uviesť základnú charakteristiku genetických algoritmov. Takisto je

cieľom práce uviesť oblasti, kde sú tieto dva nástroje úspešne využívané, prípadne v akých oblastiach bolo ich použitie preskúmané. Pre naplnenie tohto cieľa je potrebné preštudovať vedecké práce z oblasti praktickej aplikácie genetických algoritmov a popísať problematiku na základe vhodných literárnych zdrojov.

Druhým a zároveň hlavným cieľom práce, požadovaným fyzikálnym ústavom AVČR je vytvorenie programu, ktorý bude schopný aproximovať zadané vstupy podľa požiadaviek spísaných v kapitole 5. Splnenie tohto cieľa je dosiahnuteľné správnym namapovaním danej problematiky na riešenie pomocou genetických algoritmov, vhodným aplikovaním metódy najmenších štvorcov na genetické algoritmy a voľbou čo najlepších podporných overených knižníc a nástrojov pre kvalitnú implementáciu programu. Neoddeliteľnou súčasťou naplnenia hlavného cieľu je vytvorenie návodu na používanie implementovaného programu.

1.3 Predpoklady a obmedzenia práce

Základným predpokladom pri tvorbe praktickej časti práce je fakt, že vyvinutá aplikácia bude používaná vedeckými pracovníkmi, ktorí majú určité skúsenosti s aplikáciami podobného charakteru. Z toho dôvodu nie je problém nechať používateľovi väčšiu voľnosť pri parametrizácii programu, bez obáv z toho, že by po naštudovaní užívateľskej príručky neboli schopní vhodne parametre programu nastavovať. Takisto nie sú stanovené prehnané požiadavky na prílišnú dizajnérsky precíznu podobu grafického rozhrania, keďže sa jedná o aplikáciu s najväčším dôrazom na funkcionalitu. Na druhej strane to však určite neznamená automatické zníženie užívateľskej prívetivosti a kladenie dôrazu na jednoduchosť a intuitívnosť používania.

1.4 Výstupy a očakávané prínosy práce

Predpokladané výstupy práce sa podobne ako téma práce delia na teoretické a praktické výstupy. Medzi hlavné teoretické výstupy patrí:

- Rešerš prác zaoberajúcich sa praktickou aplikáciou genetických algoritmov na riešenie nielen optimalizačných úloh a využitie metódy najmenších štvorcov.
- Zaradenie genetických algoritmov do oblasti výpočtových modelov, popis základných stavebných jednotiek genetických algoritmov, možné prístupy k ich tvorbe a návrhu.
- Popis aproximácie matematických funkcií s dôrazom na metódu najmenších štvorcov.

Prakticky orientovaná časť je zameraná na implementáciu programu, kľúčové výstupy tejto časti sú:

- Analýza a návrh riešenia spolu so zmapovaním najvhodnejších vývojových nástrojov a odporučením overených knižníc vhodných na implementáciu genetických algoritmov.
- Spustiteľná aplikácia, freeware, spĺňajúca požiadavky bližšie špecifikované v kapitole 5.
- Užívateľská príručka so zrozumiteľným návodom na používanie vyvinutej aplikácie.

Originalita a prínos práce spočíva v špecifickosti zadaného problému. Na podobnej úrovni a pomocou genetických algoritmov neexistujú vo svete práce, ktoré by aproximáciu matematických funkcií, s možnosťou rozdelenia množiny vstupných bodov na viacero intervalov a následnej spojitosti výsledných funkcií riešili. Preto je implementovaný program originálnym prínosom, a bude potenciálne využívaný fyzikálnym ústavom Akadémie Vied Českej Republiky s možným rozšírením do Veľkej Británie po jazykovej úprave.

2 Práce s podobnou tematikou

Úlohou tejto kapitoly je poskytnúť prehľad potenciálneho praktického využitia genetických algoritmov ako aj možnosti aplikácie metódy najmenších štvorcov. Pre pochopenie faktov uvedených v nasledujúcich podkapitolách nie sú potrebné akékoľvek teoretické základy z popisovaných oblastí, cieľom je skôr čitateľovi priblížiť aký silný nástroj na riešenie problémov môžu genetické algoritmy byť. Pri popise je čerpané hlavne z vedeckých prác bez ohľadu na miesto vzniku, pretože daná téma nie je nijak geograficky špecifická. Spočiatku sú rozobraté práce venujúce sa aproximácií parametrov nelineárnych matematických funkcií a následne sú popísané aplikácie v širšom spektre.

2.1 Praktické aplikácie genetických algoritmov

Genetické algoritmy sú stále rozšírenejším a obľúbenejším nástrojom na získanie riešenia problému, pre ktorý postačuje riešenie blížiace sa optimu. Nasledujúci zoznam typov úloh spoločne s odkazom na práce, v ktorom boli ich aplikácie vypracované zďaleka nie je vyčerpávajúci, avšak je dostatočne názorný pre získanie prehľadu o sile genetických algoritmov.

2.1.1 Aproximácia empiricky získaných dát

Prác, ktoré sú zhodné s hlavnou témou tejto práce nebolo vypracovaných mnoho. Vo väčšine prác sa aproximáciou zaoberali len pre malý počet matematických funkcií a samotné skúmanie aproximácie nebolo cieľom práce. V [4] je popísaná aproximácia polynomiálnej funkcie s dôrazom na konverziu z bitovej reprezentácie na reprezentáciu v desiatkovej sústave, používaná vo vstavaných systémoch. Takisto aj v ostatných prácach, napr. [5], [6] venujú pozornosť len aproximácií pomocou polynomiálnej funkcie, ktorá je len z jednou z funkcií, ktorú má vyvíjaný program popísaný v tejto diplomovej práci poskytovať. V [5] je naviac spísané porovnanie viacerých aproximačných metód a verifikovaná správnosť výsledkov polynomiálnej aproximácie poskytnutých genetickým algoritmom. Veľké množstvo vedeckých prác ohľadom aproximácie sa týka polygonálnej aproximácie pomocou genetických algoritmov, ktorá sa však využíva v spracovaní obrazov a rozpoznávaní tvarov.

Podobnej problematike je venovaný výskum popísaný v [7]. V tejto práci bol hľadaný nelineárny vzťah medzi nameraným indukovaným napätím a vzdialenosťou elektromagnetického senzoru. Prínos výsledkov práce uvedenej v [7] pre túto diplomovú prácu je v overení funkčnosti a spoľahlivosti aproximácie empiricky získaných bodov nelineárnou matematickou funkciou.

Žiadna dostupná vedecká práca sa nevenuje hlavnej téme mojej práce v takom rozsahu, prípadne v podobnej špecifikácií aby dokázala minimálne z časti poskytnúť riešenie daného problému. Okrem dostupných zdrojových kódov a popisov algoritmov už vzniknuté práce neriešia aproximáciu prostredníctvom širokej škály nelineárnych matematických funkcií, nutnú spojitosť na intervaloch rozdeľujúcich namerané hodnoty určené na aproximáciu. Hlavným prínosom týchto prác teda zostáva empirické overenie spoľahlivosti aplikácie genetických algoritmov na aproximáciu nelineárnych funkcií. Tento fakt zároveň uľahčuje zložité dokazovanie validity a správnosti výsledkov, ktorému sa budem venovať v menšej miere.

Jediná práca zameraná na danú problematiku je projekt vedúceho tejto diplomovej práce, RNDr. Vladimíra Tichého, ktorý aproximoval vstupné dáta pomocou analytických metód pre jeden interval. Tento projekt sa nazýva MNC a bol fyzikálnym ústavom už využívaný, preto slúžil ako inšpirácia pre funkcionalitu tejto práce, na ktorú sú pracovníci v tomto ústave zvyknutý. Zdrojové kódy aplikácie však neboli využité. Viacej je možné sa o projekte RNDr. Tichého dočítať v [8].

2.1.2 Optimalizácia sietí

V tomto prípade sa jedná o siete v širšom poňatí. Jednou z prvých aplikácií genetických algoritmov je systém na riadenie plynovodov. Konkrétne sa jedná o udržanie adekvátneho tlaku v potrubí a zároveň minimalizáciu spotrebovanej energie. Je tu potrebné aj počítať s únikmi plynu a minimalizáciou falošných poplachov. Viacej o tejto problematike je možné dočítať sa v [9]. V [9] je popísaná aplikácia genetických algoritmov na návrh podnikovej komunikačnej siete. Topológia danej siete musela spĺňať široké spektrum požiadaviek, medzi ktorými bolo pripojenie každej podsiete viac ako jednou linkou na minimalizáciu možného zlyhania siete, obmedzenia kladené na vyvažovanie rýchlosti liniek atď. Podľa [9] bol vďaka aplikácií genetických algoritmov na daný problém vytvorený taký silný nástroj, ktorého výsledky pre návrh komunikačnej sietí neboli schopný prekonať školený inžinieri, ktorým zabralo budovanie siete mnohonásobne viac času. Možné využitie v rámci optimalizácie sietí nachádzajú genetické algoritmy pri evaluácii vzniknutej chyby v mikrovlnných komunikačných systémoch a ich následnej diagnostiky ako je popísané v [10]. Ako siete je možné brať aj množiny bodov ktoré treba počas cesty navštíviť, preto je vhodné genetické algoritmy aplikovať aj pri plánovaní cestovného plánu, s dôrazom na vyhnutie sa slabo prejazdným úsekom, alebo inými podobnými obmedzeniami.

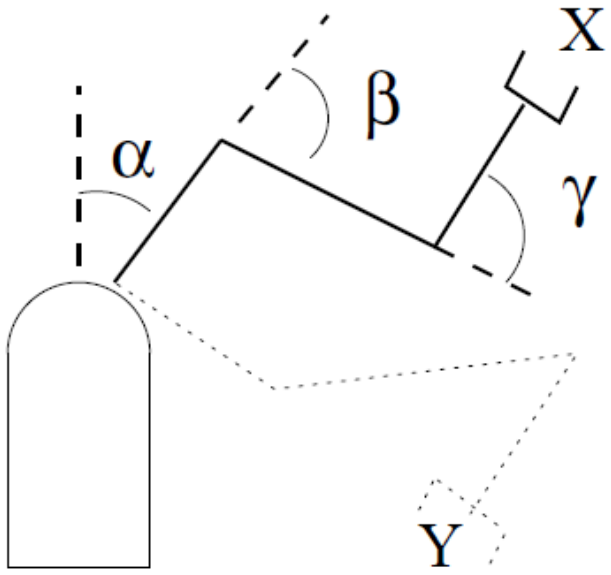
2.1.3 Tvorba rozvrhov

Genetické algoritmy nachádzajú uplatnenie v oblasti tvorby a optimalizácie rozvrhov. Praktické využitie tohto nástroja pri tvorbe rozvrhu skúšok je popísané v [11], jedná sa o vytvorenie harmonogramu skúšok tak, aby sa žiadnemu študentovi neprekrývali termíny

skúšok, aby nemal dve skúšky priamo po sebe. Testovanie opäť preukázalo, že takýto systém vytvára lepšie rozloženie skúšok ako pracovníci na toto zameraný, časovo sa je schopný ku optimálnemu výsledku dopracovať v intervale maximálne niekoľkých minút. Pri aplikácií v tomto odvetví sa systémy neobmedzujú len na tvorbu rozvrhov skúšok, v [11] je popísané ďalšie možné využitie na tvorbu časového usporiadania rozvrhov predmetov pre celú univerzitu prípadne zostavenie harmonogramu pre chod výrobnej linky alebo obsluhy strojov v továrni. Takisto sa môže jednať o systémy riešiace rozsadenie účastníkov konferencií, prípadne iných akcií na základe ľubovoľne definovaných pravidiel a preferencií.

2.1.4 Inžinierske aplikácie

Popisovaný aparát genetických algoritmov je výhodne využívaný na riešenie inžinierskych problémov, ktoré sú popísané v [12]. Medzi tieto typy úloh patrí návrh polovodičových čipov VLSI. Problém zjednodušene spočíva v stanovení takej sieti spojení medzi vstupmi a výstupmi, aby čip poskytoval požadovanú funkcionalitu s danými obmedzeniami na spotrebu energie, rýchlosť spracovania a podobne. Odlišným typom úlohy úspešne riešenej aplikáciou genetických algoritmov je v inžinierskej oblasti návrhu rotorov pre helikoptéry s nízkou vibračnosťou. Takýto systém dokáže po zadaní vstupných parametrov akými sú rozloženie hmoty, elasticita čepelí vrtule, vzduchový pretlak navrhnúť vysoko optimálnu podobu rotora s najlepšími vlastnosťami pre spoľahlivú funkcionalitu helikoptéry. Ďalšou zaujímavou oblasťou je budovanie a riadenie robotických ramien, kedy je úlohou genetických algoritmov vypočítať najvhodnejší pomer uhlov ohybu ramena pre presun z jedného miesta na druhé tak, ako je to znázornené na obrázku 2.1 [13]. Podobný princíp je možné využiť v oblasti bioinformatiky, kde sa pomocou genetických algoritmov nachádzajú štruktúry proteínov s priestorovým usporiadaním (vzájomnými uhlami medzi molekulami) s najnižšou možnou energiou [14]. Takýto postup je perspektívny pre budúce objavovanie a vývoj nových chemikálií a liekov. V oblasti inžinierskych úloh riešia genetické algoritmy ďalej tvary a zloženia nosníkov budov, turbín, zotrvačníkov alebo satelitov. Okrem návrhu tvaru a zloženia sú používané na analýzu slabín daných entít, a minimalizáciu ich kazivosti do budúcna.



Obrázok 2.1:
Uhly α, β, γ, ktoré genetický algoritmus vypočíta pre optimálny presun robotického ramena z bodu X do bodu Y

2.1.5 Definícia pravidiel

Systém predstavený v [15] objavuje a definuje pravidlá pomocou ktorých sa dokáže stíhačka vhodným manévrovaním vyhýbať riadenej strele až kým tejto strele nedôjde palivo. Stíhačka dokáže vďaka systému predpovedať dráhu letu, rýchlosť, smer a relatívnu polohu riadenej strely. Pravidlá na úspešné uhýbanie sú jednoduchého charakteru, jedná sa o príkazy typu zmeniť smer letu o 90° vpravo, pokiaľ sú zistené hodnoty o strele v určitom intervale. Systém na podobnom princípe bol upravený pre zvýšenie bezpečnosti klasických, komerčných letov a je dodávaný spoločnosťou Syllogic.

2.1.6 Predpoveď chovania cien akcií

Svoje uplatnenie nachádzajú genetické algoritmy aj vo finančnom svete, konkrétne sa jedná o oblasť obchodovania na burze. Tu môžu byť využité dvojakým spôsobom:

- budovanie vlastného automatizovaného obchodovacieho systému,
- pokročilejšia optimalizácia obchodného systému.

Pri obchodovaní na burze sa človek stretáva s veľkým množstvom ukazateľov, parametrov a premenných. Genetické algoritmy v tomto prípade pomáhajú vniesť organizáciu a ponúkajú pravidlá odvodené z hodnôt ukazateľov, ktoré vravia kedy je vhodné akcie nakúpiť a kedy naopak predať, s tým aby bola pravdepodobnosť zisku čo najvyššia. Systémy implementované na tomto princípe sú popísané napríklad v [16]. Na tomto princípe je na trhu

dostupný a podľa [16] najspoľahlivejší automatizovaný systém Adaptrade Builder. Problematika automatizovaného obchodovanie na burze s využitím genetických algoritmov je príliš rozsiahla, preto sa jej nebudem v tejto časti viac venovať. Užitočné informácie je možné nájsť v [17].

2.1.7 Vyvíjajúci sa hardware

Nielen počítačový program je schopný správať sa podľa zadaných kritérií a vyvíjať sa tak aby poskytoval čo najlepšie riešenie. V posledných rokoch sa aplikáciou genetických algoritmov na vývoj hardwaru začína používať rekonfigurovateľný hardware, ktorý dokáže samostatne reagovať na zmeny prostredia a prispôsobiť im svoje chovanie. Môže sa jednať o zamedzenie poruchovosti zariadenia pri vysokých, resp. nízkych teplotách vlastnou adaptáciou na tieto podmienky, zachovanie funkcionality v prostredí so zvýšenou rádioaktivitou a podobné inteligentné prispôsobovanie sa podmienkam okolia [18]. Postupným skúmaním v tejto oblasti je možné vyvíjať stále dokonalejšie stroje, ktoré postupne viac a viac pripomínajú ľudské chovanie.

2.1.8 Ďalšie aplikácie genetických algoritmov

Doposiaľ uvedený zoznam praktických problémov ktorých riešenie dokážu genetické algoritmy poskytnúť nie je zďaleka vyčerpávajúci. Medzi ostatné aplikácie genetických algoritmov patrí napríklad [19]:

- V automobilovom priemysle sa využívajú genetické algoritmy podobne ako v letectve na návrh čo najlepších aerodynamických tvarov vozidiel, prípadne dochádza k optimalizácii používaných materiálov na jednotlivé časti áut. Tento prístup umožnil extrémne urýchlenie procesu návrhu automobilov, keďže nahrádza zdĺhavé laboratórne experimentovanie s polymérmi a veternými tunelmi jednoducho počítačovou simuláciou.
- Genetické algoritmy sa podarilo aplikovať v lingvistickej sfére. Medzi najvýznamnejšie a najzmysluplnejšie systémy patrí STANDUP program, ktorého úlohou je vyvinutie komunikačných stratégií pre ľudí pracujúcich s deťmi s vyvinutými poruchami schopnosti komunikovať.
- Tieto algoritmy sú používaný v hernom priemysle. Existujú dva hlavné spôsoby ich využitia. Prvým je postupné učenie sa a vytváranie stratégie, pomocou ktorých sa dá hra úspešne zvládnuť, teda tie, ktoré spolupracujú s hráčom. Druhým spôsobom je využitie genetických algoritmov ako reprezentanta umelej inteligencie v hrách, a postupným učením sa stávajú objekty v hre stále viac inteligentnými.

- V oblasti bezpečnosti bývajú genetické algoritmy používané na zakódovanie citlivých dát, na druhej strane sa využívajú takisto aj na rozkódovanie zašifrovaných dát.

- V genetickej medicíne sú programy založené na báze genetických algoritmov využívané pre diagnostiku genetických porúch pacientov.

Z údajov uvedených v tejto kapitole práce vyplýva, že genetické algoritmy sú naozaj silným nástrojom pre riešenie optimalizačných problémov. Preto sa ich aplikácia ukazuje ako veľmi vhodná na riešenie zadaného problému. Zadanie problému môže byť preformulované na optimalizačný problém, kedy sú hľadané optimálne parametre nelineárnych matematických funkcií aproximujúcich zadané vstupné hodnoty.

2.2 Praktické aplikácie metódy najmenších štvorcov

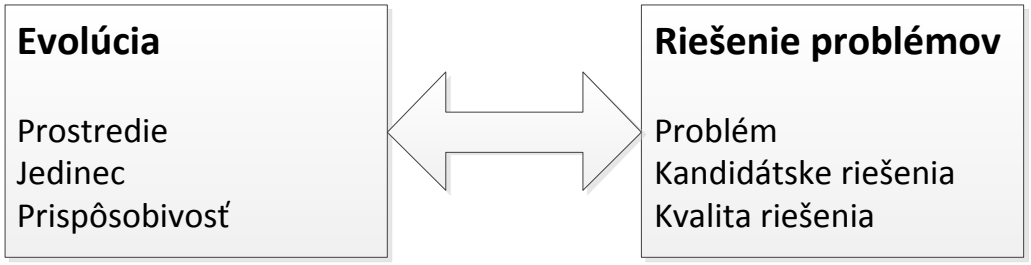
Metóda najmenších štvorcov sa ako spôsob prekladania nameraných dát na získanie aproximačnej funkcie využíva v mnohých vedeckých aplikáciách a prácach ako podporná metóda na získanie tohto predpisu. V kapitole 4 je uvedený teoretický popis tejto metódy, ktorý bližšie ozrejmuje jej veľkú rozšírenosť.

Vedecké práce zaoberajúce sa praktickými aplikáciami tejto metódy vyžadujú hlboké teoretické znalosti z iných oblastí a často nie sú reprezentatívne ani názorne pre lepšie priblíženie možností užitia metódy najmenších štvorcov. Takisto je táto metóda stará viac ako 200 rokov, tým pádom je dostatočne overená a pre účely tejto diplomovej práce nie je potrebné rozoberať jej aplikáciu v rôznych vedeckých oblastiach tak, ako to bolo potrebné pre oblasť genetických algoritmov. Pre účely práce a pre uvedenie čitateľa do problematiky je postačujúci popis tejto metódy v kapitole 4.

3 Genetické algoritmy

Táto časť práce popisuje teoretické základy genetických algoritmov, patrí sem ich zaradenie do širšieho kontextu v rámci prístupov ku riešeniu úloh. Načrtnutie ich miesta pri riešení optimalizačných úloh a následne popis základných stavebných a funkčných blokov genetických algoritmov a pravidlá a odporúčania pre ich tvorbu. Druhá kapitola demonštrovala silu genetických algoritmov ako robustného a do veľkej miery univerzálneho optimalizačného nástroja. Nedá sa však predpokladať, že bez komplexných a maximálnych znalostí o ich štruktúre a tvorbe, ich je možné efektívne aplikovať na riešenie akejkoľvek úlohy. Tento fakt je zároveň aj hlavnou motiváciou pre túto kapitolu.

Ako už bolo mierne načrtnuté v úvode práce, genetické algoritmy sú silne inšpirované Darwinovou evolučnou teóriou o prirodzenom výbere a prežití najprispôsobivejších jedincov okolitým podmienkam. Viac informácií je možné sa o Darwinovej teórii dočítať v [20]. Genetické algoritmy je možné chápať ako metaforu evolúcie pri riešení problému. Vzťah medzi evolučným procesom v prírode a genetickým algoritmom zachytáva obrázok 3.1 [21].



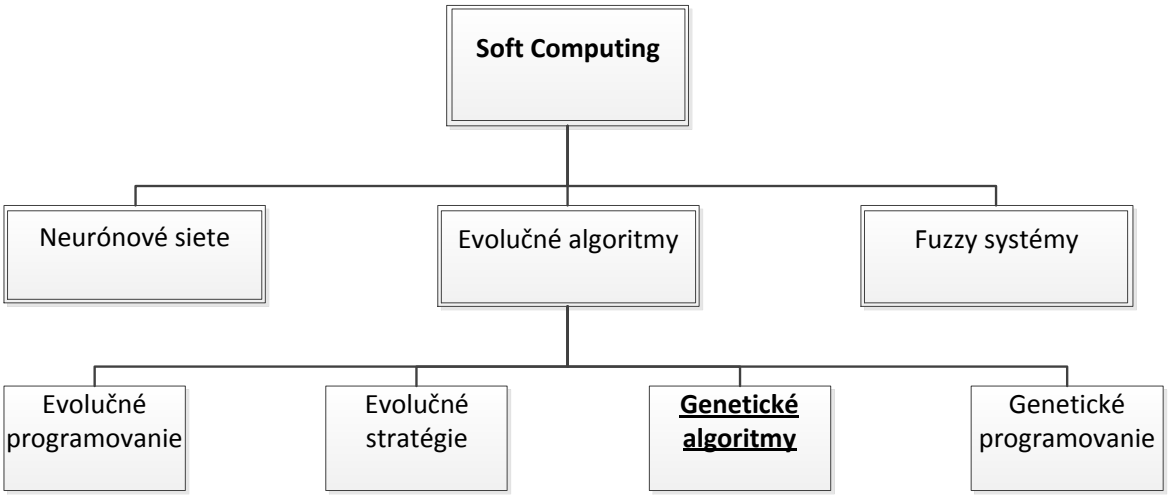
Obrázok 3.1:
Vzťah medzi evolučným procesom v prírode a riešením úloh genetickými algoritmami

V prípade genetických algoritmov je populácia zastúpená množinou potenciálnych riešení k problému, pričom problém zastupuje prostredie. Každý jedinec v populácii prejde procesom ohodnotenia jeho schopnosti zastávať optimálne riešenie. Na základe hodnotiacej funkcie je mu pridelené ohodnotenie, v zahraničnej literatúre a stále viac aj v našej, označovaná pojmom fitness. Čím lepšiu má daný jedinec reprezentujúci riešenie fitness, tým je pravdepodobnejšie, že sa bude účastniť na tvorbe ďalšej generácie riešení. Riešenia, ktoré najlepšie spĺňajú požiadavky na optimálnosť podstupujú procesy kríženia a mutácie, vďaka ktorým vznikne nová generácia riešení. Postupnou iteráciou popísaného procesu sa dospeje ku riešeniu, ktoré je najoptimálnejšie. Tento proces bude v nasledujúcich kapitolách popísaný detailnejšie.

3.1 Zaradenie genetických algoritmov

Pred samotným popisom častí genetických algoritmov je vhodné uviesť tento aparát do kontextu umelej inteligencie a definovať jeho rolu pri riešení optimalizačných úloh. Z hľadiska začlenenia do sféry umelej inteligencie patria genetické algoritmy do oblasti soft computingu.

Soft computing je odvetvie umelej inteligencie, kde je pri riešení problémov prihliadané na pravdepodobnosť, neurčitosť, sú tolerované nepresnosti a aplikuje sa všade tam, kde sú uspokojivé nie presné výsledky, ale také, ktoré sa tým presným čo najviac približujú. Hlavnou výhodou metód soft computingu sú nízke prevádzkové náklady ako aj vysoko uspokojivé výsledky, ktoré sa osvedčili počas niekoľkých desaťročí jeho používania [22]. Funkčným príkladom soft computingu je ľudská myseľ. Obrázok 3.2 znázorňuje pozíciu genetických algoritmov v rámci soft computingu [21].

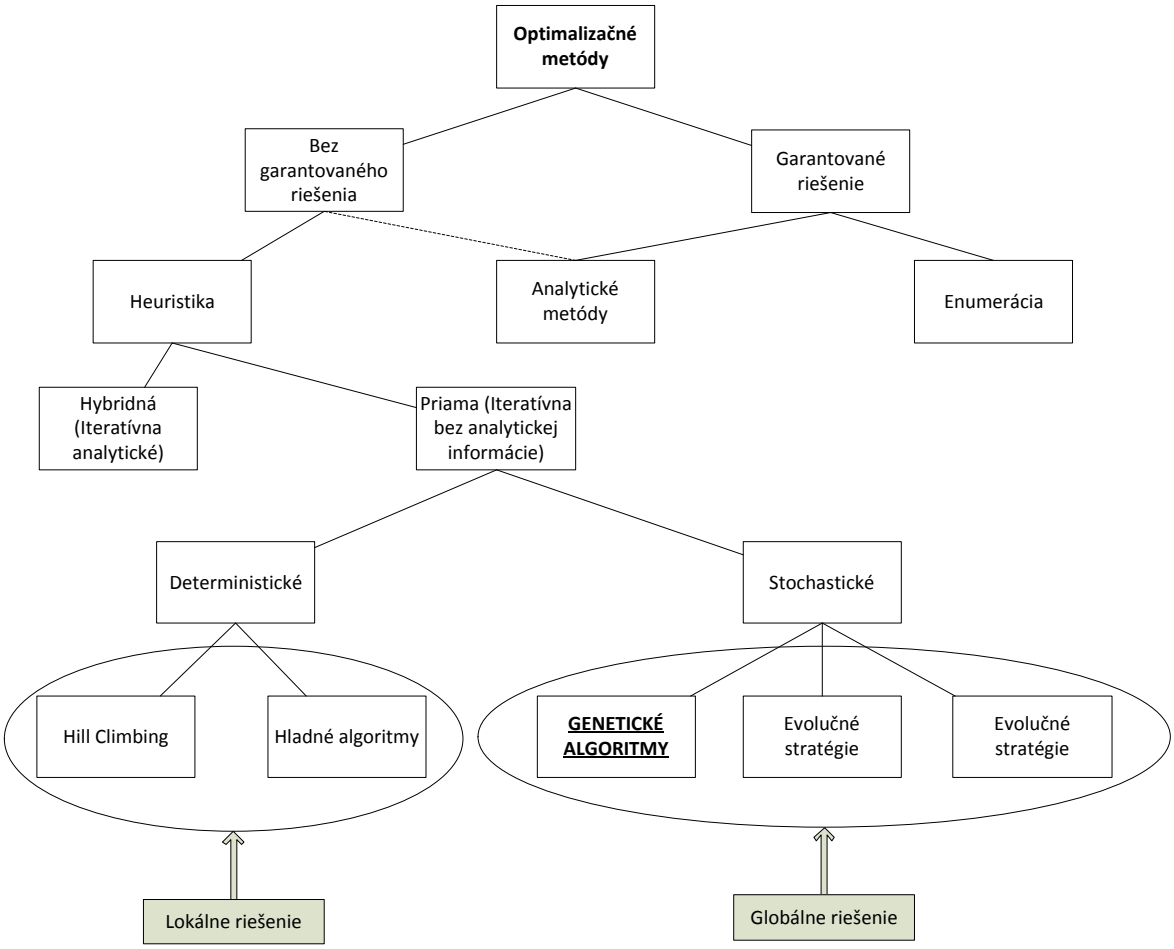


Obrázok 3.2:
Zaradenie genetických algoritmov do oblasti soft computingu

Z obrázku je zrejmé, že genetické algoritmy tvoria podmnožinu evolučných algoritmov. Tie sú na úrovni s neurónovými sieťami a fuzzy systémami. Na prvej úrovni sa volí systém podľa typu úloh ktoré je potrebné vyriešiť. Ako bolo spomenuté, evolučné algoritmy slúžia pre optimalizáciu. Neurónové siete nachádzajú široké využitie pri práci s nekompletnými, prípadne porušenými dátami, kedy sa na vstupnom vzorku neurónová sieť natrénuje, a zjednodušene povedané, je schopná naučiť sa dopracovať sa k riešeniu. Fuzzy systémy slúžia na riadenie mechanizmov, kde sa počíta s neurčitosťou prípadne s nejasnosťou zadania. Skutočnú silu nástrojov soft computingu využívajú nástroje, ktoré vyššie popísané tri systémy vhodne kombinujú. Neurónové siete na návrh funkcionality, fuzzy systémy na riadenie a evolučné algoritmy na optimalizáciu funkčnosti vzhľadom na konkrétne potreby [22].

Evolučné algoritmy zahŕňajú aparáty, ktoré sa snažia napodobniť prírodné procesy evolúcie. Genetické algoritmy naviac pridávajú elementy genetiky ako sú gény, genómy, chromozómy. Pri voľbe medzi nástrojmi evolučných algoritmov je vďaka ich širokej miere parametrizácie v literatúre [21] uvádzané, že voľba medzi nimi závisí hlavne na osobných skúsenostiach a preferenciách autora. Každý z nich dokáže poskytnúť suboptimálne riešenia na zhodné problémy a v mnohých prípadoch je vzdialenosť poskytnutého riešenia od optima len otázkou správneho nastavenia parametrov. Popisu iných metód ako genetických algoritmov sa preto v tejto práci nebudem venovať.

Po zaradení genetických algoritmov do konkrétnej oblasti umelej inteligencie je žiaduce a vhodné zaradiť ich do kontextu existujúcich optimalizačných metód. Dôležité je uvedomiť si, že genetické algoritmy nie sú vhodné na riešenie každého typu optimalizačnej úlohy. Obrázok 3.3, prevzatý z [23], zobrazuje ich zaradenie do sféry optimalizačných metód.



Obrázok 3.3:
Zaradenie genetických algoritmov do kontextu optimalizačných metód

Z obrázku 3.3 je možné vyčítať, že existuje pomerne veľké množstvo optimalizačných nástrojov. Každý z nich sa hodí na iný typ úlohy. Je potrebné ozrejmiť si, nielen ktorý typ úloh je najlepšie riešiť pomocou genetických algoritmov, ale takisto aj aký typ úloh je nimi zbytočné riešiť, pretože ich použitie by bolo prehnané, resp. nevhodné. Z optimalizačných

techník s garantovaným riešením je najpoužívanejšou enumerácia. Enumerácia sa používa pri optimalizovaní úloh s malým rozsahom riešeného problému. To znamená, pokiaľ existuje napríklad 5 možností hodnôt parametrov, je použitie iných metód ako enumerácie zbytočné. Pri hľadaní extrémov jednoduchších matematických funkcií bývajú najčastejšie používané analytické metódy (prvá a druhá derivácia, Newtonova metóda). V prípade, že nie je požadované globálne najlepšie riešenie úlohy, zväčša je to v prípade vysokého počtu možných riešení, je použitá niektorá z heuristických metód. Heuristika poskytne nejaké, prijateľné, riešenie, o ktorom však nie je známe či je globálne najlepšie. Priame heuristické techniky sú aplikované v takom prípade, keď nie sú známe analytické informácie (schopnosť derivácie, spojitosť). Pri takýchto úloh sú k dispozícií deterministické metódy, prvou sú hladné algoritmy, ktoré poskytnú prvé vhodné riešenie. Druhou deterministickou metódou je metóda hill climbing, ktorej hlavnou slabinou je možnosť uviaznutia v lokálnom extréme. Genetické algoritmy patria medzi priame, stochastické heuristické metódy [23]. Ak je potrebné riešiť nejaký problém, zvyčajne je možné tento problém previesť na optimalizačný. Po zvážení jeho povahy, rozsiahlosti, teda počtu vyhľadávaných parametrov sa v prípade uspokojenia sa so suboptimálnym riešením volia genetické algoritmy. Hlavným problémom týchto algoritmov je vysoká časová náročnosť, kedy dopracovanie sa k riešeniu môže zabrať aj niekoľko týždňov, resp. mesiacov. Ďalšou nevýhodou je poskytnutie riešenia, ktoré sa k optimálnemu len približuje. Prax však ukázala, že nachádzané riešenia sú vo veľkom zastúpení globálne optimálne, v najhoršom prípade suboptimálne [21].

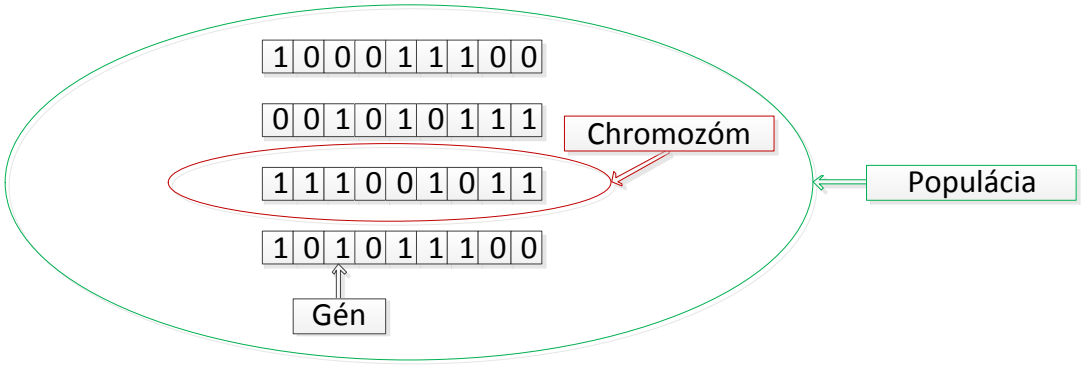
3.2 Základné pojmy

Genetické algoritmy preberajú pomenovania svojich komponent z biologického procesu evolúcie. V popise ich fungovania budú tieto pojmy často používané, v nasledujúcej časti sú jednotlivé pojmy vysvetlené tak, ako sú chápané genetickými algoritmami.

Základné pojmy s ktorými bude v texte operované sú:

- **gén** – základná stavebná jednotka chromozómu, je to symbol nad používanou abecedou (napr. pri binárnej reprezentácií 0 alebo 1), atomický nositeľ informácie,
- **alela** – konkrétna hodnota génu,
- **chromozóm** – často označovaný ako indivíduum, skladá sa z génov, je nositeľom riešenia problému, v niektorých nástrojoch pre implementáciu genetických algoritmov sa používa označenie **genóm,** v prípade reprezentácie jedinca ako štruktúry zloženej z viacerých čísiel sa aj táto štruktúra označuje ako chromozóm a jednotlivé čísla, ktoré obsahuje sú gény,

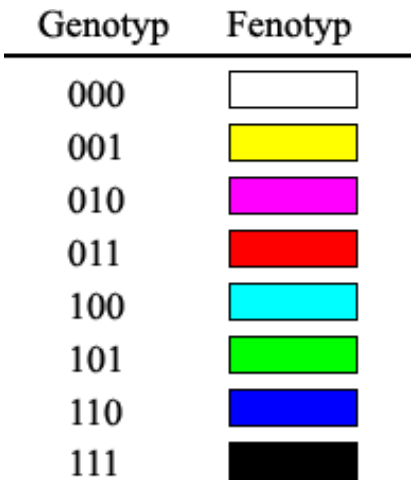
- **populácia** – konečná množina chromozómov, reprezentujúcich riešenia daného problému, počet jedincov v populácii býva spravidla konštantný, obrázok 3.4 zobrazuje vzťah medzi doposiaľ menovanými pojmami,



Obrázok 3.4:
Zobrazenie vzťahu medzi génom, chromozómom a populáciou v oblasti genetických algoritmov

- **genotyp** – zakódovaný tvar riešenia (napr. binárny reťazec),

- **fenotyp** – dekódovaná forma riešenia, spojitosť medzi genotypom a fenotypom demonštruje obrázok 3.5 [24], tento obrázok zachytáva zobrazenie genotypu na fenotyp pre kódovanie farby modelom CMY, kde je prítomnosť farby vo fenotype zachytená symbolom „1“ a jej absencia symbolom „0“,



Obrázok 3.5:
Zobrazenie genotypu farby pomocou modelu CMY na fenotyp

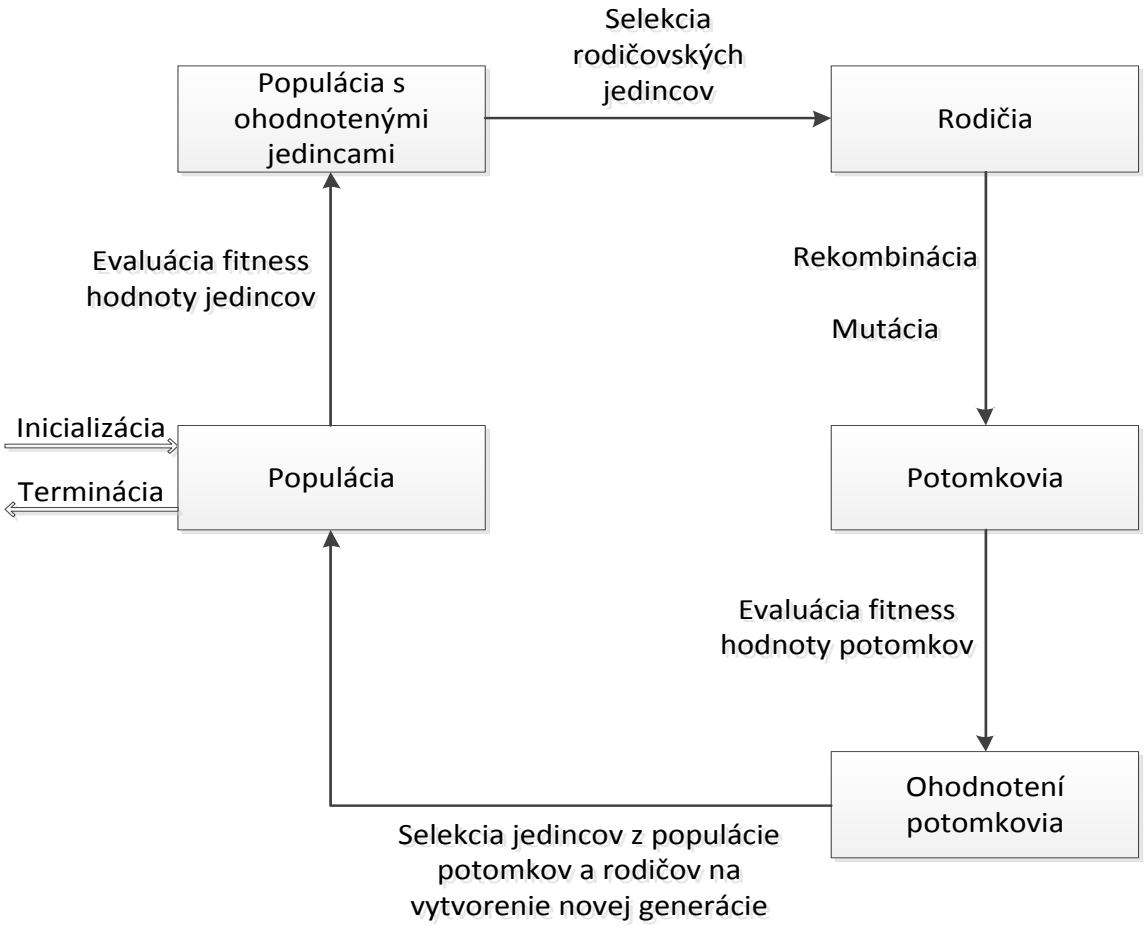
- **generácia** – iteratívny charakter genetických algoritmov spôsobuje obmenu populácie v každom kroku, generácia je potom populácia v konkrétnom kroku,

- **fitness** – vlastnosť jedinca, kvantifikovaná miera priblíženia sa optimálnemu riešeniu, na základe tejto hodnoty sú jedinci ďalej vyberaní do tvorby novej generácie

- **fitness funkcia** – nazývaná aj ohodnocovacia funkcia, táto funkcia prideľuje každému jednotlivcovi populácie jeho fitness hodnotu.

3.3 Štruktúra genetického algoritmu

Genetické algoritmy majú vo všetkých aplikáciách jednu zhodnú vec, a tou je ich štruktúra. Bez ohľadu na to na aký problém sú aplikované, zakaždým sa riadia overenou a presne stanovenou základnou kostrou krokov, kde sú niektoré etapy voliteľné, iné spravidla prebehnú vždy. Síce je ich štruktúra pomerne ustálená, programátor má mnoho možností, ako pomocou parametrov jednotlivé etapy modifikovať, prípadne úplne z behu vynechať. Táto modifikácia sa však zvyčajne uskutočňuje po vytvorení genetického algoritmu a na základe experimentovania s empirickými výsledkami je s parametrami manipulované. Treba poznamenať, že k tomuto nemusí dochádzať len zásahom z vonka, existujú špeciálne typy genetických algoritmov, ktoré sú schopné modifikovať svoje parametre samé. Na obrázku 3.6 je zachytená základná štruktúra genetického algoritmu [21].



Obrázok 3.6:
Základná schéma genetického algoritmu

Prebiehajúce procesy sú na obrázku 3.6 znázornené šípkou, vznikajúce entity sú v obdĺžniku. Po počiatočnej inicializácii vzniká prvá generácia populácie jedincov. Tieto sú následne ohodnotené fitness funkciou. Na základe fitness hodnoty dochádza ku selekcii tých jedincov, ktorý sa zúčastnia na tvorbe novej množiny potomkov za pomoci rekombinácie a kríženia rodičovských génov. Po vzniku populácie potomkov sú jedinci tejto množiny ohodnotený fitness hodnotou a podľa jej hodnoty dôjde k súťaži medzi rodičmi a potomkami o to, ktorí jedinci vytvoria novú generáciu indivíduí. Procesy ku ktorým tu dochádza sú detailnejšie popísané v nasledujúcich podkapitolách.

3.3.1 Reprezentácia jedincov

Spôsob reprezentácie jedincov je súčasťou procesu inicializácie. Voľbou reprezentácie je však ovplyvnená každá ďalšia operácia v priebehu genetických algoritmov. Jedná sa o zvolenie množiny hodnôt, ktoré môžu chromozómy nadobúdať. Zároveň je voľba reprezentácie jedincov aj prvým krokom v návrhu genetických algoritmov, návrhu sa podrobnejšie venuje kapitola 6.4. Informácie v tejto časti sú prevzaté z [21].

Veľmi častou, najjednoduchšou a najstaršou formou reprezentácie jedincov je binárne zakódovanie. V tomto prípade je každý chromozóm reťazcom binárnych hodnôt, ukážka takto zakódovaného chromozómu je na obrázku 3.4. Historicky sa vyskytlo mnoho algoritmov, ktoré nasilu používali binárnu reprezentáciu bez ohľadu na charakter riešeného problému. Pri binárnej reprezentácií je nutné zvoliť dĺžku binárneho reťazca, a ako budú jeho hodnoty interpretované. Teda spôsob akým sa genotyp prevedie na fenotyp vhodný pre riešenie. Príklad prevodu genotypu a fenotypu binárneho reťazca je uvedený na obrázku 3.5. Ďalšou podmienkou je zabezpečenie toho, aby dekódovanie binárnej reprezentácie poskytlo len validné riešenia daného problému a naopak, že všetky možné riešenia je možné reprezentovať pomocou binárneho zakódovania. Binárna reprezentácia je vhodná pre prirodzený proces kríženia a mutácie. Binárna reprezentácia sa nepoužíva len v prípadoch, kedy je fenotyp riešenia logická hodnota „0“ a „1“, ale bývajú jeho pomocou kódované aj celé, prípadne reálne čísla.

Druhým typom reprezentácie je celočíselné kódovanie. Toto býva používané v prípadoch, kedy rôzne gény môžu nadobúdať viac ako dve hodnoty. Typickým príkladom je problém určenia smeru cesty, kde celočíselné konštanty 0,1,2,3 zastupujú smer pohybu S, V, Z, J.

Použitie reálnych hodnôt pri reprezentácie jedincov je najvhodnejšie v takom prípade, ak je pátrané po výsledku na spojitom intervale a nie na diskrétnom obore hodnôt. Takýmto príkladom je hľadanie reálnych koeficientov funkcie pre aproximáciu matematických funkcií.[25]

Celočíselná a aj reálna reprezentácia jedincov môže byť vhodným kódovaním nahradená binárnou reprezentáciou. Tu však môžu nastať problémy s vysokou časovou náročnosťou programu z dôvodu vysokej požadovanej presnosti reprezentácie, ktorá zvyšuje dĺžku bi-

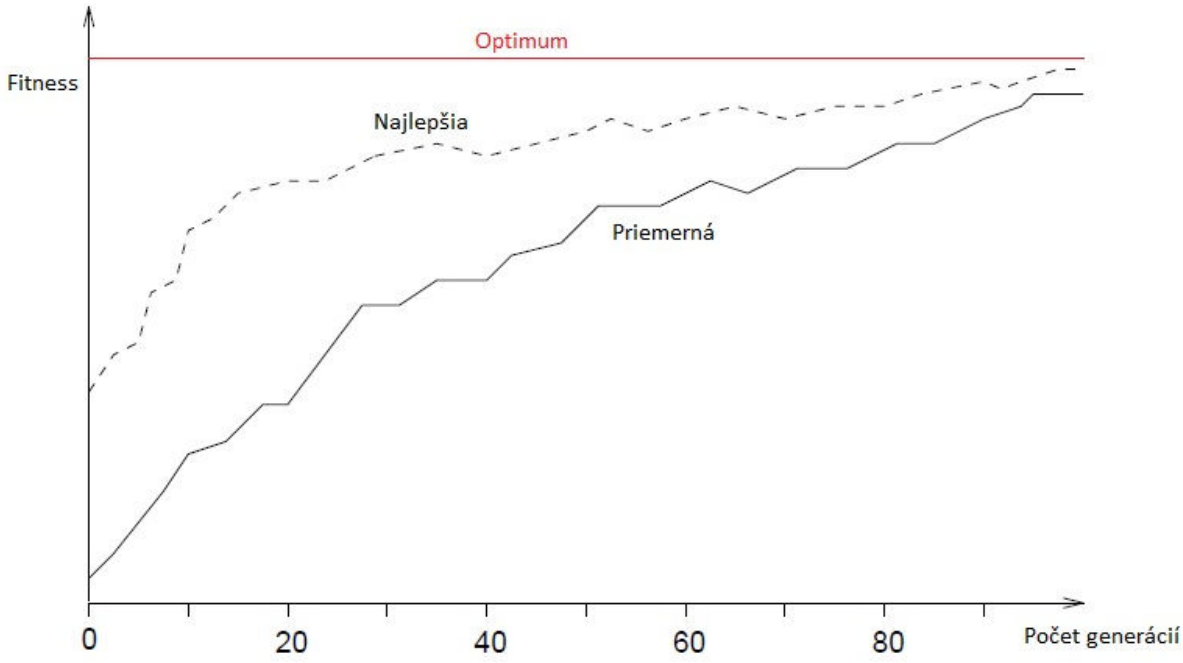
nárneho reťazca. Prax ukazuje, že voľba reprezentácie by sa mala odvíjať hlavne od typu úlohy a rýchlosti a vhodnosti prevodu genotypu na fenotyp riešenia v danej reprezentácii.

3.3.2 Ohodnotenie jedincov

Ohodnocovacia funkcia slúži na vypočítanie fitness hodnoty každého jedinca. Fitness hodnota slúži na to, aby bolo možné posúdiť, ktorý jedinci budú zvolení pre tvorbu ďalšej generácie jedincov. Alegoricky s prírodným procesom evolúcie, jedinci s lepším ohodnotením, vyššou fitness hodnotou, sa s vyššou pravdepodobnosťou zúčastnia na tvorbe ďalšej generácií a prežijú dlhšie. Ohodnocovacia funkcia musí pre každého jedinca vracať takú hodnotu, ktorá je merateľná, zmysluplná a porovnateľná s fitness hodnotami ostatných jedincov.

Veľkým problémom genetických algoritmov je vysporiadanie sa s obmedzujúcimi podmienkami. Takou podmienkou môže byť napríklad rovnosť súčtu hľadaných premenných danej hodnote. Ak nie je možné túto podmienku zakomponovať do hodnotiacej funkcie, zavádza sa penalizačná funkcia. Penalizačná funkcia znižuje fitness hodnotu jedinca v závislosti od toho, do akej miery narušuje obmedzujúce podmienky.

Obrázok 3.7 [26] ukazuje typický priebeh fitness hodnoty v závislosti na počte generácií. Prerušovanou čiarou je zobrazená najlepšia fitness konkrétnej generácie, plná čiara znázorňuje priemernú hodnotu fitness. Červená čiara symbolizuje optimum, ku ktorému postupne hodnota fitness funkcie konverguje. Z obrázku je zrejmé, ako sa postupnou evolúciou zlepšuje priemerná fitness populácie, vďaka mechanizmom popísaným v tejto kapitole.



Obrázok 3.7:
Typický priebeh fitness funkcie

3.3.3 Selekcia rodičov

Pridelenie fitness hodnoty každému jedincovi slúži ako kritérium pre voľbu jedincov na tvorbe ďalšej generácie riešení. Prístupov ako týchto rodičovských jedincov vyberať existuje niekoľko, v tejto časti sú popísané tie najpoužívanejšie metódy. Popis prístupov je prevzatý z [23] a [26].

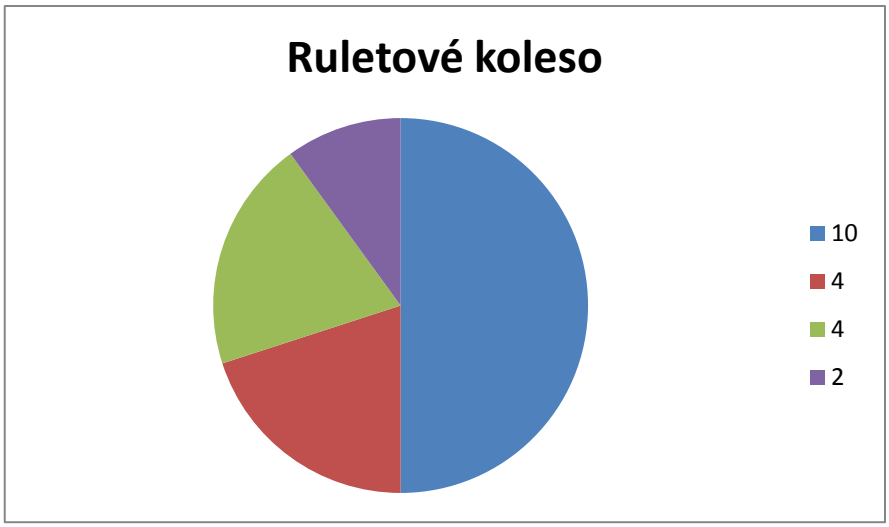
Proporcionálna selekcia

Pravdepodobnosť selekcie jedinca je závislá na pomere jeho absolútnej hodnoty fitness ku sume absolútnych hodnôt fitness ostatných jedincov. Čím vyššiu fitness jedinec má, tým je pravdepodobnejšie, že bude zvolený ako rodič. Pravdepodobnosť výberu i-tého jedinca je daná vzťahom:

$$p_i = \frac{f_i}{\sum_{j=0}^{N} f_j} \tag{3.1}$$

Kde p_i je pravdepodobnosť výberu i-tého jedinca, f_i je fitness hodnota tohto jedinca, N je celkový počet jedincov (veľkosť populácie), a suma v menovateli zodpovedá súčtu fitness hodnôt celej populácie.

Implementačne je proporcionálna selekcia riešená pomocou tzv. ruletovej selekcie. Obsah plochy rulety kruhového tvaru zodpovedá súčtu fitness hodnoty jedincov celej populácie. Jednotlivé kruhové výseče na nej reprezentujú fitness hodnoty jedincov. Po roztočení rulety je zrejmé, že s najvyššou pravdepodobnosťou na nej padne jedinec s najväčšou plochou kruhovej výseče, resp. fitness hodnotou. Tento proces je znázornený na obrázku 3.9 [28].



Obrázok 3.8:
Rozdelenie ruletového kolesa pre hodnoty fitness 10, 4, 4, 2

Proporcionálna selekcia je síce najpoužívanejším mechanizmom, hlavne z dôvodu jednoduchej implementácie a poskytovanie relatívne vhodných výsledkov, má však hneď niekoľko obmedzení:

- prirýchla konvergencia riešenia, výnimočný jedinci s vysokou fitness oproti zvyšku populácie rýchlo ovládnu celú populáciu,
- selekčný tlak, pokiaľ sa fitness v populáciu líšia o veľmi nízke hodnoty môže riešenie konvergovať neúmerne dlho

V snahe minimalizovať negatívne vlastnosti proporcionálnej selekcie boli navrhnutý iné metódy selekcie.

Selekcia usporiadaním

Pri použití tejto metódy sú jedinci usporiadaný na základe ich fitness. Následne je pravdepodobnosť ich voľba závislá len od poradia v usporiadaní, a nie od veľkosti ich fitness funkcie. Mapovanie poradia jednotlivca na pravdepodobnosť výberu býva spravidla prevádzané lineárnym, alebo exponenciálnym usporiadaním. Pre lineárne usporiadanie je pravdepodobnosť výberu i-tého jedinca v zoradenej populácii nasledujúca, kde s je parameter a $1{,}0 < s \leq 2{,}0$ a N je veľkosť populácie :

$$P_{lin}(i) = \frac{2-s}{N} - \frac{2i(s-1)}{N(N-1)}$$

(3.2)

Pre exponenciálne usporiadanie to je:

$$P_{exp}(i) = \frac{1-e^{i}}{c}$$

(3.3)

Parameter *c* je normalizačný faktor a volí sa v rozmedzí 0 < c < 1. Tento selekčný algoritmus patrí medzi najlepšie algoritmy čo sa týka eliminácie nežiaducich obmedzení [21].

Turnajová selekcia

Predchádzajúce dve metódy výberu sa spoliehali na znalosť fitness celej populácie. Hlavne v prípadoch príliš rozsiahlych populácii, prípadne pri použití paralelizmu je túto informáciu obtiažne získať. V takých prípadoch sa volí turnamentová selekcia. Jeho princíp je v náhodnom výbere dvoch jedincov do turnaju, jedinec s vyššou fitness je následne zvolený do procesu tvorby novej generácie. Tento proces môže byť modifikovaný, napríklad pridaním viacerých úrovní turnaja, prípadne kombináciou ruletového výberu jedincov a následnou turnajovou selekciou medzi nimi. Hlavnou devízou tohto výberu je absencia nutnosti zoradenia populácie a jednoduchosť vlastnej selekcie.

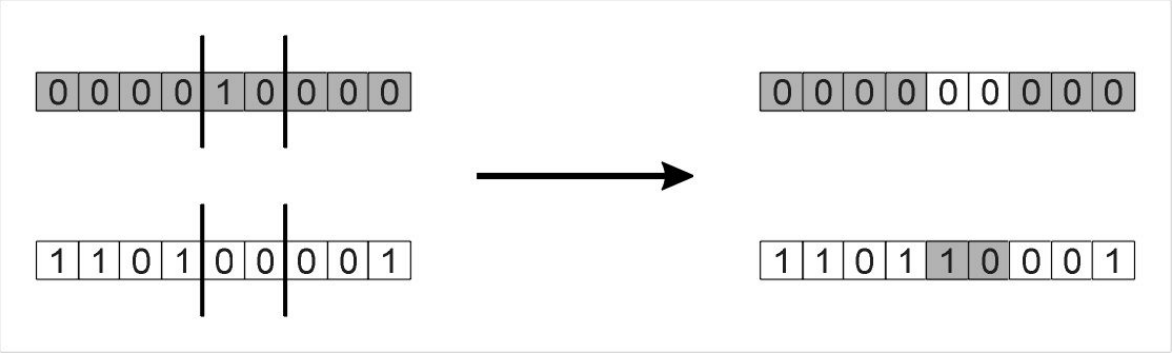
Elitizmus

Táto metóda sama o sebe nie je spôsobom selekcie, avšak eliminuje jedno potenciálne nebezpečenstvo každej selekčnej metódy. Tým je možnosť nezahrnutia najsilnejších jedincov medzi rodičovskú populáciu. Elitizmus zabezpečuje to, aby selekčný algoritmus zakaždým zahrnul istý, zvolený počet najsilnejších jedincov danej generácie buď do tvorby potomkov, alebo rovno do nasledujúcej generácie [29].

3.3.4 Rekombinácia

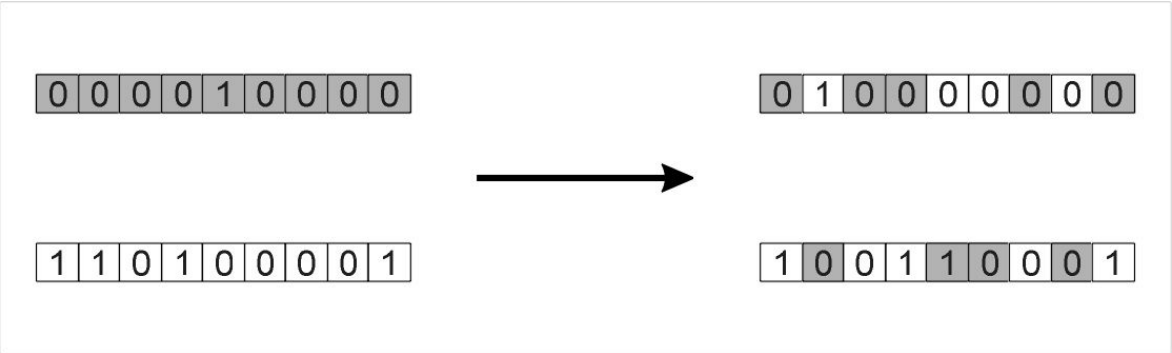
Iným názvom pre tento proces používaným v literatúre je aj kríženie. Po výbere jedincov, ktorý majú najlepšiu fitness, teda reprezentujú najoptimálnejšie riešenie, dochádza k procesu rekombinácie genetického materiálu medzi rodičmi a vzniku potomkov, ako novovzniknutých nositeľov riešenia. Do tohto procesu vstupujú vždy dvaja rodičia a vznikajú dvaja potomkovia. Zvyklosťou je, že ku rekombinácii dochádza s relatívne vysokou pravdepodobnosťou (20%-80%).

V praxi najčastejšie používaným a najviac názorným postupom kríženia je jednobodové a viacbodové kríženie. Tento mechanizmus vychádza z biologickej analógie kde sa rekombinácia uskutočňuje v jednom, alebo vo viacerých bodoch chromozómu. V prípade *n*-bodového kríženia dochádza ku výmene *n* úsekov rodičovských chromozómov. Vzniknú tak dvaja potomkovia, ktorí sú zložený z častí chromozómov oboch rodičov. Tento proces je znázornený na obrázku 3.9 [29]. Na ľavej strane obrázku sa nachádzajú rodičia, na pravej potomkovia vzniknutý dvojbodovou rekombináciou rodičovských chromozómov.



Obrázok 3.9:
Príklad n-bodového kríženia pre n=2

Na druhej strane ako *n*-bodové kríženie stojí uniformné kríženie. Toto namiesto skladania chromozómu potomkov po skupinách génov pristupuje ku samostatným génom, a na základe istej pravdepodobnosti rozhodne, z ktorého rodiča bude daný gén pridelený potomkovi. Pravdepodobnosť na základe ktorej sa určí dominantný rodič pre daný gén môže vyplývať napríklad z fitness rodičov, prípadne môže byť celkom náhodná. Uniformné kríženie zachytáva obrázok 3.10 [29].



Obrázok 3.10:
Príklad uniformného kríženia

Dve doposiaľ popísané metódy patria medzi najpoužívanejšie v prípade binárnej reprezentácie jedincov (kapitola 3.3.1). Pri použití celočíselnej, resp. reálnej reprezentácie existujú dva možné prístupy [21]:

- Na kríženie oboch reprezentácii použiť vyššie popísaný aparát, a teda uniformné alebo n-bodové kríženie, s tým že gény nebudú obsahovať len binárne hodnoty ale

všetky číslice desiatkovej sústavy. Tento prístup sa odporúča hlavne pri celočíselnej reprezentácii. Pri reálnej sa z dôvodu rozdelenia čísla na celú a desatinnú časť tento prístup začína zanechávať a prechádza sa na druhý, aritmetický spôsob.

- Pre chromozómy reprezentované reálnymi číslami je najvhodnejšie použiť aritmetické kríženie. Teda previesť rekombináciou oboch rodičov tak, aby z nich nejakou aritmetickou operáciou vznikli dvaja potomkovia. V praxi sa najčastejšie používa nasledujúci aritmetický vzťah 3.4, R_i je rodič, P_i zastupuje potomka a α je reálny parameter, pre ktorý platí že $0 \leq \alpha \leq 1$:

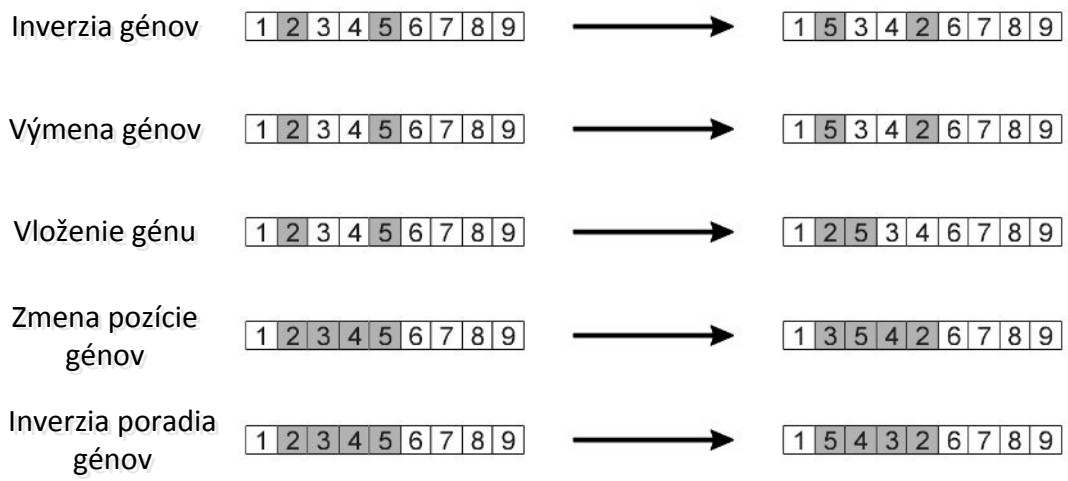
$$P_1 = \alpha * R_1 + (1 - \alpha)R_2$$

$$P_2 = (1 - \alpha)R_1 + \alpha * R_2 \qquad (3.4)$$

3.3.5 Mutácia

Význam mutácie v genetických algoritmoch je zhodný s jej biologickým významom, a tým je vnesenie elementu náhody do génového fondu. Do tejto operácie vstupuje jeden chromozóm a vystupuje z nej jeden potomok. Mutácia sa spravidla prevádza na novovzniknutých potomkoch tak, ako je tomu aj v prírode. Zväčša ku mutácií dochádza s veľmi nízkou pravdepodobnosťou (0,05% - 1%), na rozdiel od kríženia. Príliš veľká pravdepodobnosť mutácie môže znehodnotiť prirodzený vývoj populácie, a naopak príliš malá pravdepodobnosť nemusí vniesť dostatok nových impulzov do vývoja.

Existuje rozsiahle množstvo mutačných operátorov, teda spôsobom, ako môže k mutácii dôjsť. Tie sú podobne ako pri rekombinácii závislé od zvolenej reprezentácie (kapitola 3.3.1). Pre binárne kódovanie je najčastejšie používanou mutácia pomocou invertovania hodnoty náhodného génu, prípadne viacerých génov. Ďalej pre celočíselnú a binárnu reprezentáciu sú používané mutácie výmenou, kedy sa dva náhodne vybrané gény v chromozóme vymenia. Ďalším spôsobom je zmena pozície náhodne zvoleného génu, zmena pozície viacerých génov v rámci istej časti chromozómu a inverzia poriadia génov v časti chromozómu. Všetky tieto mutačné operátory sú znázornené na obrázku 3.11 [29].



Obrázok 3.11:
Príklady rôznych metód mutácie

Podobne ako pri rekombinácii je pre reálnu reprezentáciu chromozómov najvhodnejšie použiť aritmetické metódy mutácie. Medzi aritmetické mutácie patrí inkrementácia, resp. dekrementácia o náhodnú hodnotu, ktorá sa spravidla nachádza v rozmedzí 0% - 10% mutovaného chromozómu.

3.3.6 Selekcia jedincov pre novú generáciu

Táto časť býva často nazývaná aj obnova populácie. Po vzniku potomkov krížením a mutáciou, existujú dve množiny, množina rodičov a množina potomkov. Z týchto množín je potrebné zvoliť presný počet jedincov na obnovu populácie. Existujú tri prístupy [21]:

- úplná obnova populácie, kde je rodičovská populácia plne nahradená potomkami,
- čiastočná obnova populácia, kde jeden najvhodnejší potomok nahradí najslabšieho rodiča,
- zmiešaná obnova populácie, ktorá je zároveň aj najpoužívanejšou, kedy sa určité percento populácie rodičov (20%-50%) nahradí najlepšími potomkami.

Výber, ktorého rodiča nahradiť, resp. ktorého potomka zvoliť na obnovu populácie môže prebiehať na základe dvoch kritérií:

- selekcia s ohľadom na vek rodiča, ktorej základom je neprihliadanie na fitness hodnotu jedinca, ale na počet iterácií genetického algoritmu, ktorých sa daný rodič zúčastnil, po dosiahnutí vopred stanoveného počtu iterácií je tento rodič zvolený ku vyradeniu z populácie a potomok, ktorý tohto rodiča nahradí je buď vybraný náhodne, alebo na základe jeho fitness,

- selekcia s ohľadom na fitness, kde rodičia s najnižšou fitness sú nahrádzaní potomkami s vyššou fitness, táto metóda obnovy populácie môže byť vhodne implementovaná pomocou ruletovej alebo turnamentovej selekcie, popísaných v kapitole 3.3.3.

3.4 Postup návrhu genetického algoritmu

Pri praktickej aplikácii genetických algoritmov je treba dodržať niekoľko základných krokov, po ktorých splnení bude vytvorený genetický algoritmus, a jeho správne využitie bude len otázkou kvalitnej implementácie vo zvolenom programovacom jazyku. Jednotlivé kroky návrhu mapujú body spísané v predchádzajúcej podkapitole 3.3.

Pri návrhu a tvorbe genetického algoritmu je teda potrebné dodržiavať nasledujúce kroky, tieto body tvoria otázky, ktoré potrebuje programátor pred samotnou implementáciou genetického algoritmu vyriešiť:

1. Zvolenie vhodnej reprezentácie problému.
2. Inicializácia počiatočnej populácie.
3. Spôsob ohodnocovania jedincov populácie.
4. Selekcia párov rodičovskej populácie.
5. Metódy genetických operácií kríženia a mutácie.
6. Spôsob obnovy populácie a ukončenie algoritmu.

Praktická náplň týchto bodov je uvedená ďalej v texte, pri návrhu genetického algoritmu, ktorý rieši otázku aproximácie matematických funkcií.

4 Metóda najmenších štvorcov

Pri potrebe aproximácie empiricky nameraných dát je jednou z najstarších a najpoužívanejších metód práve metóda najmenších štvorcov. Najčastejšie sa metóda používa na lineárnu aproximáciu, teda preloženie nameraných dát priamkou. Časté sú však aj prípady nelineárnej aplikácie metódy, napríklad aproximácia pomocou paraboly, hyperboly, polynomiálnej, prípadne exponenciálnej funkcie atď. Z dôvodu veľkého rozšírenia a širokej všeobecnej znalosti tejto metódy je v tejto práci popísaná len stručne. Poznatky zhrnuté v tejto kapitole sú prevzaté z [31].

Historicky základy metódy najmenších štvorcov položil Carl Friedrich Gauss v roku 1795 [30]. Pri aplikácii metódy na lineárny aj nelineárny prípad je základná myšlienka zhodná. Je potrebné nájsť predpis takej funkcie *f*, pre ktorú platí, že druhá mocnina y-ových vzdialeností aproximovaných bodov je minimálna. Podstatu tejto metódy ilustruje pravá časť obrázka 4.1 [32].

Pre lineárny prípad platí, že priamka $y=a*x + b$ je analytickým vyjadrením priamky preloženej súborom bodov $[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]$ pomocou metódy najmenších štvorcov, pokiaľ pre koeficienty *a, b* platí:

$$a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i\, y_i\,,$$

$$a\sum_{i=1}^{n} x_i + bn = \sum_{i=1}^{n} y_i$$

(4.1)

Pre nelineárny prípad aproximácie polynomiálnej funkcie druhého stupňa $y= a*x^2+b*x+c$ metódou najmenších štvorcov musí pre koeficienty platiť nasledujúci vzťah [33]:

$$a + b\sum_{i=1}^{n} x_i + c\sum_{i=1}^{n} x_i^2 + = \sum_{i=1}^{n} y_i$$

$$a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x_i^3 + = \sum_{i=1}^{n} x_i\, y_i$$

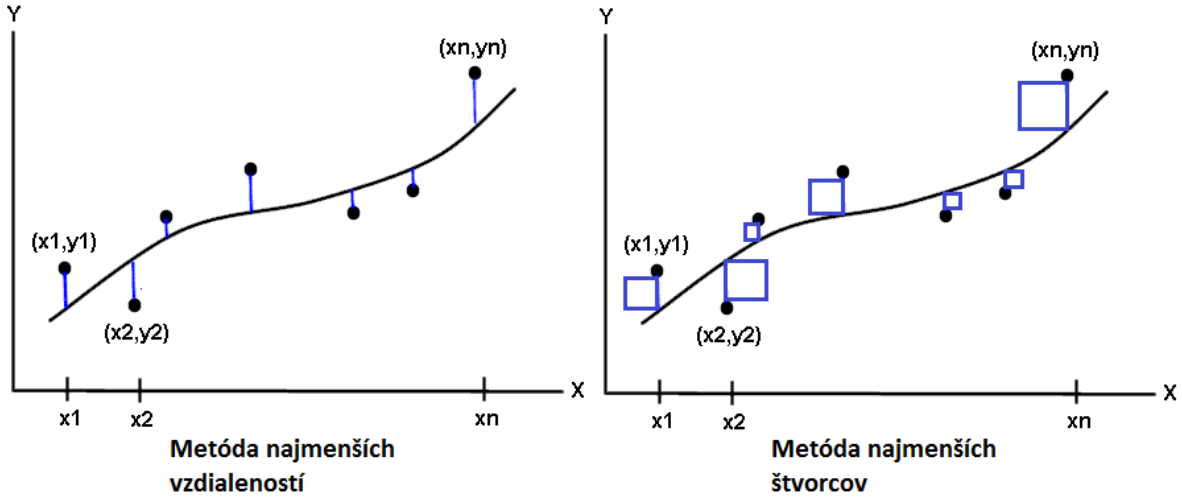
$$a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i^3 + c\sum_{i=1}^{n} x_i^4 + = \sum_{i=1}^{n} x_i^4\, y_i$$

(4.2)

Na základe vzťahu 4.2 je možné odvodiť rovnice pre výpočet koeficientov pre polynómy n-tého stupňa. Pre ostatné nelineárne prípady taktiež existujú rovnice, ktoré sú prehľadne uvedené v [32]. Popis jednotlivých nelineárnych prípadov aproximácie a vzťahov na získanie hodnoty parametrov nelineárnych funkcií by presahoval rámec práce. Úlohou práce je dospieť ku aplikácií metódy najmenších štvorcov na genetické algoritmy aj pre nelineárne prípady, táto aplikácia je popísaná v nasledujúcej kapitole, kde je pri riešení zadanej úlohy metóda najmenších štvorcov používaná ako ohodnocovacia funkcia prideľujúca fitness hodnotu jedincom.

4.1 Metóda najmenších vzdialeností

Riešenie daného problému aproximácie pomocou genetických algoritmov nevyžaduje počítanie pomocou derivácií. Preto som zvolil možnosť voľby metódy aproximácie medzi aproximáciou pomocou druhej mocniny y-ovej vzdialenosti bodu od priamky, alebo absolútnej hodnoty vzdialenosti bodu od priamky. Dôvodom je fakt, že aproximácia pomocou metódy najmenších štvorcov je ľahko ovplyvniteľná náhodnými a nie často sa vyskytujúcimi chybami merania. Teda empiricky získané hodnoty, ktoré sa veľmi líšia od priemeru ostatných hodnôt a môžu byť považované za anomálie neovplyvnia natoľko výslednú aproximovanú funkciu pri použití metódy najmenších vzdialeností, pretože sa počíta len s absolútnou hodnotou tejto vzdialenosti a nie jej druhou mocninou. Obrázok 4.1 [32] ilustruje rozdiel medzi dvoma použitými metódami.



Obrázok 4.1:
Ilustrácia princípu použitých matematických metód

5 Požiadavky na aplikáciu

Táto kapitola má za cieľ na základe požiadaviek definovať prípady užitia, ktoré budú pokrývať užívateľskú interakciu s programom.

Ako už bolo spomenuté v úvodnej časti práce, požiadavky na aplikáciu boli definované Fyzikálnym ústavom Akadémie vied ČR. Tieto požiadavky sú natoľko všeobecné, že aplikácia bude môcť byť potenciálne využívaná väčším množstvom vedeckých ústavov, teda nejedná sa o konkrétny problém, ktorý potrebuje riešiť len jeden ústav.

Aplikácia má za úlohu aproximovať empiricky namerané hodnoty matematickými funkciami, ktoré budú volené užívateľom. Výsledok je potrebné zobraziť formou grafu. Výsledný graf má zodpovedať diagramu prechodov rôznych látok medzi skupenstvami. To znamená, že namerané body môžu zodpovedať termodynamickým veličinám, tlaku, teploty a objemu a výsledný graf zachytáva závislosti medzi nimi. Napríklad závislosti objemu od teploty a pod. V grafe nemusia byť zobrazené len závislosti medzi termodynamickými veličinami, môže sa jednať napríklad o závislosť percentuálneho obsahu uhlíka v železe na teplote železa. Všetky namerané hodnoty sú dostupné, potrebné je získať predpis funkcií, ktoré by toto chovanie látky čo najlepšie popisovali.

Keďže sa jedná o prechod látky medzi troma skupenstvami, jej fyzikálne, teda aj termodynamické, charakteristiky sa na základe formy skupenstva menia, a tým sa mení aj podoba krivky, ktorá popisuje závislosti termodynamických veličín. Preto je nutné, aby aplikácia dokázala aproximovať hodnoty v troch intervaloch, pričom pre každý interval môže byť volená rôzna matematická funkcia, ale zároveň aby tieto funkcie boli spojité. Spojitosť je nutná z toho dôvodu, že zmena fyzikálneho skupenstva neprináša diskrétne skoky, ale charakteristika danej látky je spojitá.

5.1 Nefunkčné požiadavky na systém

Medzi nefunkčné požiadavky patria požiadavky na grafické užívateľské rozhranie, dostupnosť a bezpečnosť.

Grafické užívateľské rozhranie (GUI)

Výsledná aplikácia bude mať formu grafickej aplikácie. Z dôvodu, že program budú používať vedeckí pracovníci, je žiaduce aby bol vývoj sústredený hlavne na poskytnutie požadovanej funkcionality a nie sú požadované prehnané grafické, respektíve dizajnérske prvky. Prvky GUI majú byť intuitívne a aby nebola práca s nimi problematická. Hlavnou požiadavkou na GUI je jeho prehľadnosť a aby v rámci jednej obrazovky ponúkalo všetky možnosti programu. Požadované je menu aplikácie, ktorým bude možné s programom ma-

nipulovať a ovládať jeho činnosť. Každá položka menu by mala byť spustiteľná pomocou klávesovej skratky. Okrem ovládania programu bude GUI ponúkať aj výstup programu, viď. 5.2. Pri voľbe prvkov GUI je vhodné sa inšpirovať už používaným softvérom MNC [8].

Dostupnosť

Potenciálni užívatelia aplikácie sú okrem pracovníkov Fyzikálneho ústavu AV ČR vedeckí pracovníci po celom svete, preto je nutné, aby bola aplikácia dostupná online. Keďže existuje možnosť, že nie všetky počítačové stanice koncových užívateľov sú nepretržite pripojené na internet, je treba rátať s tým, že aplikácia by mala byť spustiteľná aj na takýchto počítačoch. Požadovaná forma distribúcie a dostupnosti je však prostredníctvom umiestnenia aplikácie na web.

Bezpečnosť

Pri používaní aplikácie vedeckými pracoviskami bude dochádzať ku zadávaniu súkromných údajov, respektíve nameraných dát danými pracoviskami do aplikácie. Tento fakt zvyšuje nároky na bezpečnosť aplikácie, pretože vstupné dáta aplikácie, tvoria privátne vlastníctvo konkrétneho vedeckého ústavu.

5.2 Funkčné požiadavky na systém

V tejto časti práce sú spísané požiadavky na funkcionalitu systému. Základná a hlavná funkcionalita, ktorú systém musí poskytovať je aproximácia zadaných vstupných bodov pomocou niekoľkých matematických funkcií spojitých na troch intervaloch, ktorých hranice zadá užívateľ. Konkrétna podoba naplnenia funkčných požiadaviek opäť môže byť inšpirovaná programom RNDr. Tichého, MNC [8].

Vstup programu

Program bude schopný načítať ako vstup hranice jednotlivých intervalov vymedzujúcich oblasť pre aproximáciu matematickými funkciami. Ďalej bude možné načítať pre jednotlivé intervaly namerané body, ktoré budú zadané pomocou x-ových a y-ových súradníc. Medzi ďalšie vhodné, nie však povinné vstupné parametre programu patrí údaj o základnom rozsahu zobrazenia súradnicových osí a aproximačná funkcia pre jednotlivé intervaly. Interaktívne zadanie bodov, nie je z dôvodu vyššej nepresnosti takejto formy vkladania vstupu, nutne požadované. Zadávateľ je zvyknutý na zadávanie vstupu formou parametrického, respektíve konfiguračného súboru, preto bude vhodné takúto formu vkladania vstupných hodnôt ponechať.

Výstup programu

Ako už bolo spomenuté, aplikácia je požadovaná s grafickou podobou. Hlavnou časťou grafického výstupu bude dvojrozmerný graf, zobrazujúci výsledné matematické funkcie aproximujúce zadané vstupné body. Tento graf bude takisto zachytávať zadané body ako aj hranice intervalov, zodpovedajúcich jednotlivým skupenstvám skúmanej látky. Okrem výstupu formou grafu je vyžadovaný aj analytický predpis každej výslednej aproximačnej funkcie.

Užívateľská interakcia

Okrem manipulácie s programom formou vstupného konfiguračného súboru, musí byť pred spustením výpočtu umožnené užívateľovi pomocou komponent GUI nastaviť nasledujúce parametre:

- konkrétny typ matematickej funkcie na aproximáciu zadaných vstupných bodov pre každý interval,
- voľba metódy aproximácie, tzn. buď aproximácia pomocou metódy najmenších štvorcov, alebo pomocou metódy najmenších vzdialeností,
- parametre v závislosti na použitom aparáte na aproximáciu, napr. počet iterácií, presnosť výsledného riešenia.

Typy aproximačných matematických funkcií

Pri vytváraní a implementácií zoznamu ponúkaných matematických funkcií na aproximáciu bude najvhodnejšie inšpirovať sa zoznamom matematických funkcií poskytovaných už používaným programom MNC.

Podporná funkcionalita

Medzi ďalšiu požadovanú funkcionalitu, ktorá nepatrí do základnej a nijak neovplyvní podstatu výstupu programu patrí:

- možnosť skopírovania výstupných dát do schránky tak, aby následnou klávesovou skratkou CTRL+V v textovom, prípadne obrázkovom, editore bolo možné vložiť do tohto editora ako výstupné analytické predpisy funkcií tak aj výsledný graf aplikácie,
- voľba rôznych zobrazovaných komponent na výslednom grafe, užívateľ bude môcť zvoliť, ktorý zo zobrazovaných prvkov chce z výstupného grafu odstrániť, respektíve pridať,
- zobrazenie krátkeho návodu na používanie aplikácie.

Chybové hlásenia

Program bude schopný vyhodnotiť vzniknuté chybové stavy a tieto náležitou a zrozumiteľnou chybovou hláškou oznámiť užívateľovi. Medzi požadované kontroly patrí:

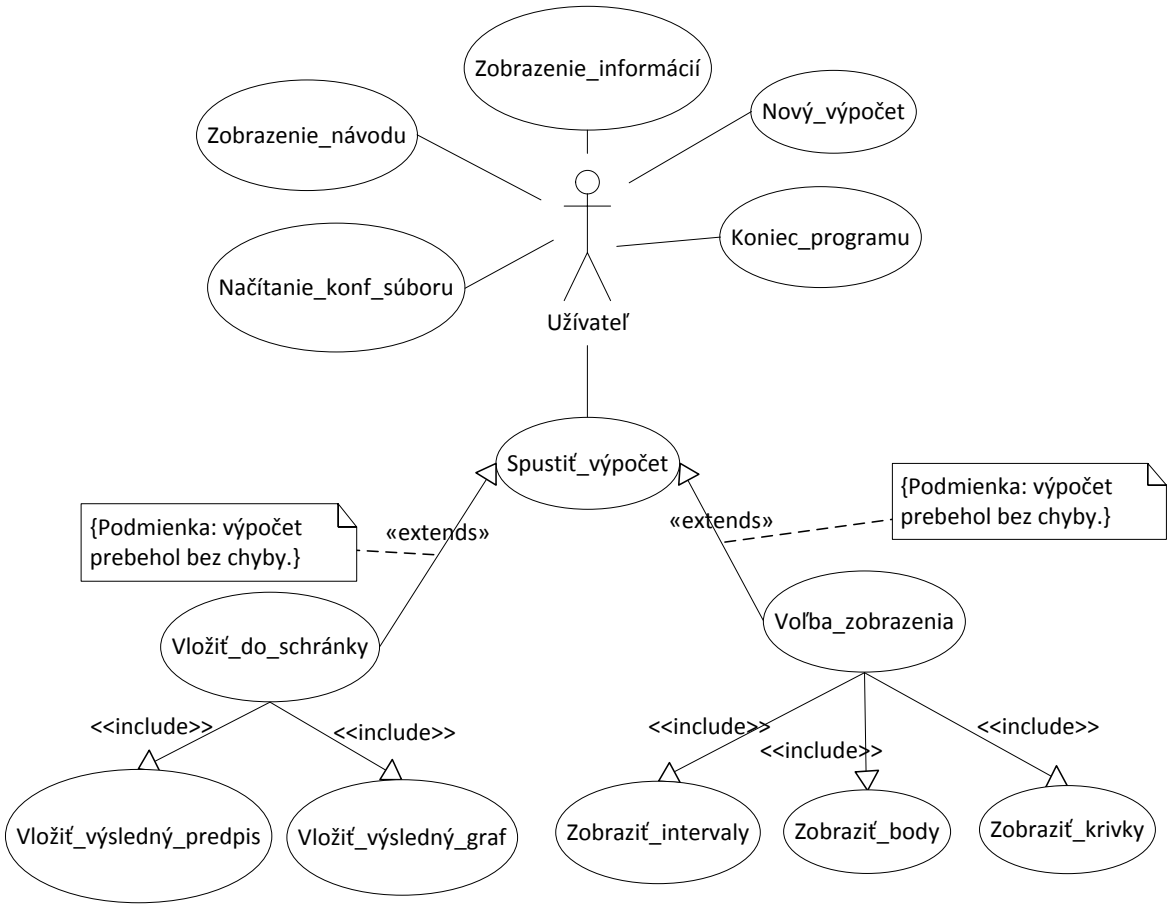
- kontrola korektnosti vstupu (neprekrývanie sa zadaných intervalov, náležitosť vstupných bodov do intervalov...),
- náležitosť vstupných bodov do definičného oboru aproximujúcej matematickej funkcie.

Aktéri systému

Aplikácia podporuje len jeden typ aktéri, ktorým je užívateľ s plnými právami na prevádzanie všetkých úkonov aplikáciou poskytovaných.

5.3 Prípady užitia systému

Prípady užitia zobrazené formou use case diagramu sú zachytené na obrázku 5.1. Popis jednotlivých prípadov užitia nasleduje ďalej v texte pod obrázkom.



Obrázok 5.1:
Use case diagram systému

Spustiť výpočet

Po zadaní vstupu je možné spustiť výpočet pomocou tlačítka na to určeného. Pri nedodržaní všetkých náležitostí potrebných na spustenie výpočtu program túto skutočnosť užívateľovi ohlási. Po úspešnom prebehnutí výpočtu sa na grafickom výstupe aplikácie zobrazí grafická podoba aproximovaných kriviek ako aj analytický zápis týchto kriviek.

Vložiť do schránky

Po úspešnom prebehnutí výpočtu užívateľ pomocou položiek hlavného menu, alebo klávesových skratiek je schopný vložiť do schránky výsledný graf vo forme obrázku a analytický predpis výsledných aproximovaných matematických funkcií buď pre jednotlivé intervaly zvlášť, alebo pre všetky intervaly.

Voľba zobrazenia

Po úspešnom prebehnutí výpočtu užívateľ zvolí pomocou položiek menu alebo klávesových skratiek tie elementy v grafu, ktoré chce aby boli zobrazené.

Nový výpočet

Pokiaľ užívateľovi nevyhovuje podoba aproximovaných kriviek, môže si pre už načítaný konfiguračný súbor spustiť pomocou položiek menu, alebo klávesovej skratky nový výpočet pre už zadaný vstup. Pokiaľ chce previesť výpočet pre nové vstupné dáta, stačí načítať nový konfiguračný súbor.

Koniec programu

Pomocou klávesovej skratky, položkami menu alebo kliknutím na krížik užívateľ program ukončí.

Zobrazenie návodu

Užívateľ pomocou položiek menu alebo klávesovou skratkou zobrazí základný návod na používanie aplikácie.

Zobrazenie informácií

Užívateľ pomocou položiek menu alebo klávesovou skratkou zobrazí základné informácie o verzii programu, autorovi a dôvodu vzniku programu.

6 Analýza a návrh riešenia

Táto kapitola je zameraná na návrh konkrétneho riešenia. Jedná sa hlavne o abstraktný návrh formou diagramov a náčrtov. Táto časť popisuje nasledujúce kroky analýzy, ktoré boli prevedené v pred implementačnej fáze projektu:

1. Analýza a návrh technológie – výber najvhodnejšej podoby finálneho produktu, s ohľadom na jeho možnosť distribúcie.
2. Analýza a návrh grafického rozhrania – náčrt podoby užívateľské rozhrania vyplývajúci z požiadaviek špecifikovaných v predchádzajúcej kapitole.
3. Logický návrh – analýza a návrh potrebných modulov pre kompletnú funkčnosť aplikácie, tvar vstupných a výstupných dát atď.
4. Návrh genetického algoritmu – jadro práce, popis komponent genetického algoritmu, ktorý rieši aproximáciu nelineárnych matematických funkcií.

Vypracovanie potrebných náležitostí v každej fáze je kľúčovým pre úspešnú implementáciu finálneho produktu. Voľba použitých a alternatívnych nástrojov na implementáciu riešenia je popísaná v ďalšej kapitole.

6.1 Analýza a návrh technológie

Fundamentálnou časťou analýzy je výber správnej technológie pre implementáciu. Do úvahy treba brať jednak možnosť distribúcie aplikácie, jej prístupnosť a bezpečnosť. Okrem týchto predpokladov, som samozrejme musel brať ohľad na moje praktické znalosti jednotlivých technológií. Od tohto výberu sa následne odvíja každý ďalší krok analýzy, preto je nutné, dôkladne premyslieť použitú technológiu. V každom prípade bude na vývoj použitý objektovo orientovaný prístup, kvôli tejto voľbe som zúžil možnú podobu finálnej aplikácie na tieto možnosti:

- webová aplikácia,
- spustiteľná .exe aplikácia,
- spustiteľný skript.

Na rozhodnutie medzi týmito tromi prístupmi som zvolil spôsob spísania ich kladov a záporov, na ich základe a na miere naplnenia očakávaných predpokladov som zvolil najvhodnejšieho kandidáta.

Webová aplikácia

Pri zvolení tejto technológie by bola aplikácia vytvorená pomocou prostriedkov pre vývoj webu. Výhody a nevýhody tejto voľby sú uvedené v tabuľke 6.1.

Tabuľka 6.1: *Výhody a nevýhody tvorby programu ako webovej aplikácie.*

Webová aplikácia	
Klady	**Zápory**
+ možnosť rýchlej implementácie	- bezpečnostné riziko odosielania súkromných nameraných dát po sieti
+ platformová nezávislosť	- nutnosť neustáleho pripojenia na internet v prípade používania
+ relatívne dobrá dostupnosť	- absencia vhodných podporných nástrojov pre genetické algoritmy
+ jednoduchá distribúcia	- nevýhody súvisiace s podstatou webových aplikácií (hosting, prerušenie poskytovania služby a znemožnenie jej ďalšieho používania, výpadky a nedostupnosť služby...)
+ odpadá nutnosť inštalácie softvéru	- čiastočne nevyhovujúce zadaniu, ktoré nepriamo požaduje aplikáciu ako freeware
+ jednoduchšia aktualizácia a oprava chýb	
+ minimálne požiadavky na užívateľovo technologické vybavenie (internetový prehliadač)	
+ uľahčené testovanie len na jednej platforme	
+ rýchla tvorba užívateľského rozhrania	

Webovú aplikáciu som volil ako prvú vhodnú možnosť, keďže moja znalosť programovacieho jazyka Ruby a webového frameworku Rails je na vysokej úrovni, programovanie by bolo jednoduché. Jej výhody vyplývajú zo samotnej podstaty webových aplikácií, ako je uvedené v tabuľke 6.1 výhody prevažujú nad nevýhodami.

Spustiteľná aplikácia

Touto možnosťou sa myslí aplikácia s koncovkou .exe, ktorej podstata ju predurčuje len na použitie so systémom Windows. Tento fakt však nie je považovaný za veľké negatívum, pretože zadanie nevyžaduje multiplatformové použitie. Vývoj by takisto nebol problém z dôvodu mojej znalosti niekoľkých programovacích jazykov poskytujúcich vytvorenie spustiteľnej aplikácie. Tabuľka 6.2 popisuje pozitíva a negatíva tejto voľby.

Tabuľka 6.2: *Výhody a nevýhody tvorby programu ako spustiteľnej aplikácie.*

Spustiteľná aplikácia	
Klady	**Zápory**
+ možnosť rýchlej implementácie + vysoká bezpečnosť pri práci so súkromnými dátami + možnosť spustenia aj na počítačoch nepripojených do siete + distribúcia pomocou zverejnenia na webe + vysoké množstvo vývojových prostredí pre rýchlu tvorbu GUI + široká podpora vývoja genetických algoritmov + rozšírené nástroje tretích strán na bežné úkony + autorov prehľad v oblasti programovacích jazykov vhodných pre túto voľbu + stála dostupnosť služby	- nutnosť inštalácie programu - distribúcia programu spoločne s externými knižnicami - riešenie len pre jednu platformu

Pre implementáciu spustiteľnej aplikácie sa naskytá viacero možných a vhodných programovacích nástrojov, ich popis a voľbu obsahuje kapitola 7.

Spustiteľný skript

Táto možnosť patrila skôr medzi teoretické možnosti implementácie, pokiaľ by som sa chcel držať vývoja v programovacom jazyku Ruby. Túto voľbu som zvažoval hlavne kvôli vysokej sile a vyššej miere abstrakcie príkazov skriptovacích jazykov oproti klasickým programovacím jazykom.

Prvou možnosťou by bolo vytvoriť skript, ktorý by užívateľ pred každým použitím spustil, spôsob spustenia skriptu by však aj pri očakávanom užívateľovi formátu vedeckého pracovníka nemusel byť jednoduchý. Takisto aj tvorba grafického rozhrania by bola značne obmedzená len na finálny výsledok, čo je v tomto prípadne nepraktické a nežiaduce.

Druhá možnosť využitia skriptovacieho jazyka spočíva v konverzii kódu do spustiteľnej aplikácie. Na internete je dostupných niekoľko systémov ponúkajúcich takéto riešenie. Po hlbšom naštudovaní problematiky som dospel k záveru, že takáto konverzia je silne nesta-

bilná a tým pádom nevhodná pre vývoj zadanej aplikácie. Rozhodnutie zostávalo len medzi webovou aplikáciou a spustiteľnou aplikáciou.

Voľba technológie

Po analýze kladov a záporov webovej a spustiteľnej aplikácie som dospel k záveru, že najvhodnejšou a bezproblémovou voľbou bude daný program vyvinúť a distribuovať ako spustiteľnú aplikáciu.

Webovú aplikáciu som nezvolil hlavne z dôvodu nízkej bezpečnosti a nutnosti prenosu nameraných súkromných dát po sieti, čo by sa niektorým vedeckým pracoviskám užívajúcich aplikáciu nemuselo zdať ako prijateľný postup. Ďalším dôvodom bola absencia kvalitných nástrojov pre vývoj genetických algoritmov pre technológiu Ruby on Rails, v ktorej by bola webová aplikácia vyvíjaná.

Pozitíva spustiteľnej aplikácie ďaleko presahujú jej negatíva. Tieto negatíva k tomu môžu byť aj do značnej miery minimalizované. Hlavnou nevýhodou je jedno platformové zameranie, toto však zodpovedá zadaniu, ktoré nepožaduje multiplatformové riešenie. Pre prípad že by sa v budúcnosti vyskytla požiadavka na rozšírenie použitia na inú platformu, budem v ďalšej analýze voliť také nástroje, ktoré je možné použiť aj na iných, rozšírených, operačných systémoch. Rozšírenie o použitie na ďalšiu platformu by potom bolo len otázkou rekompilácie aplikácie na úrovni zdrojového kódu pre danú platformu. Pre pokročilejšieho užívateľa je samozrejme aj možnosť dodania zdrojových kódov s následnou kompiláciou pre zvolenú platformu na jeho strane.

6.2 Analýza a návrh grafického rozhrania

Veľmi dôležitou súčasťou aplikácie je prehľadné užívateľské rozhranie. Táto súčasť aplikácie je dôležitá hlavne pre užívateľa, pretože hlavný výstup programu bude práve grafický výstup. Ku rozloženiu jednotlivých komponent, charakteru a výslednej podobe užívateľského rozhrania som dospel analýzou požiadaviek na finálny program.

Z dôvodu charakteru cieľového užívateľa, ktorým je vedecký pracovník, bude grafické rozhranie skôr konzervatívneho charakteru, avšak nebude to na úkor minimalizovania funkčnosti. Farebne bude aplikácia ladená nerušivými farbami, hlavne odtieňmi šedej.

Grafické užívateľské rozhranie bude tvorené jednou obrazovkou, kde hlavné rozloženie komponent zachytáva obrázok 6.1.

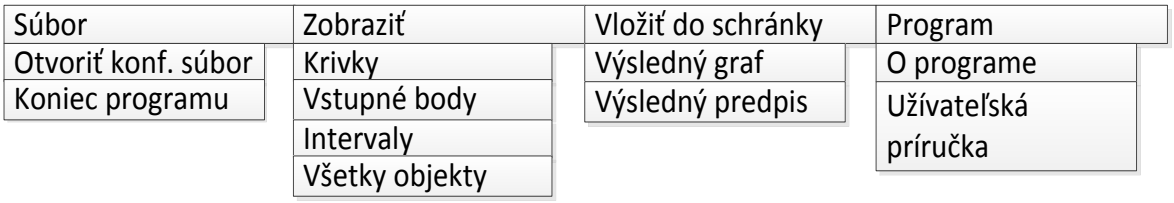
Obrázok 6.1:
Základný náčrt grafického užívateľského rozhrania

Schéma rozloženia jednotlivých komponent bude vo výslednom programe zodpovedať tej navrhnutej na obrázku 6.1, detailne bude obsah načrtnutých častí volený a upravovaný počas implementácie. Modifikácie rozhrania budú volené na základe konzultácie s vedúcim práce, prípadne pripomienkami koncového užívateľa. Je však niekoľko bodov, ktoré musí každá časť z obrázku 6.1 obsahovať. Oblasti, kde budú položky na manipuláciu s programom, sú na obrázku vyplnené svetlo šedou farbou, oblasti výstupu sú naopak tmavošedé.

Menu

Základná podoba menu je znázornená na obrázku 6.2. Táto schéma sa bude takisto ako celé grafické užívateľské prostredie meniť a prispôsobovať ešte v priebehu vývoja.

Jednotlivé položky menu zodpovedajú prípadom užitia popísaných v kapitole 5. Niektoré položky budú podľa vhodnosti implementované ako zaškrtávacie, pokiaľ to charakter akcie, ktorú reprezentujú, vyžaduje.



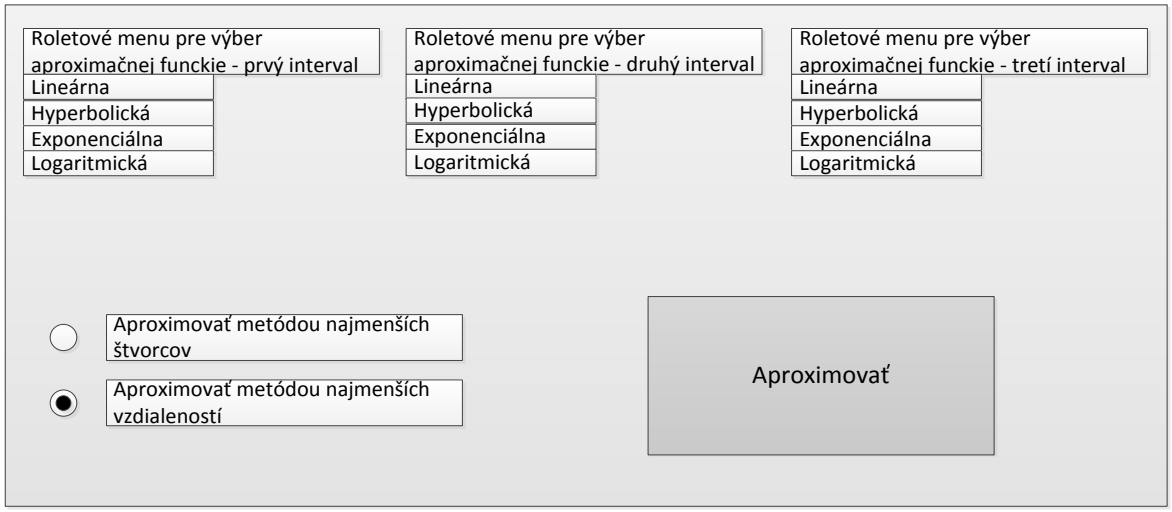
Obrázok 6.2:
Základný náčrt menu aplikácie

Hlavný grafický výstup

Plocha hlavného grafického výstupu bude obsahovať dvoj-dimenzionálny graf. Na ňom budú vyobrazené body, ktoré je potrebné aproximovať, bude tu vyobrazené rozdelenie intervalov vstupných bodov a takisto krivky zodpovedajúce aproximovaným výsledným funkciám. Zobrazený graf bude možné priblížiť aj oddialiť. Rozdelenie hlavného grafického výstupu je znázornené na obrázku 6.1 pomocou dvoch prerušovaných čiar.

Panel pre užívateľskú interakciu

Spoločne s menu ponúka táto časť grafického rozhrania užívateľovi možnosť ovládať chovanie programu. Tento panel bude obsahovať také prvky, ktoré programu umožnia naplnenie požadovanej funkcionality popísanej v predchádzajúcej kapitole. Zároveň však musí byť dostatočne prehľadný a intuitívny. Prvotná schéma, ktorej sa bude držať implementácia programu je zakreslená na obrázku 6.3.



Obrázok 6.3:
Základný náčrt panelu pre užívateľské ovládanie programu

Umiestnenie jednotlivých komponent na užívateľskom panely sa môže prispôsobovať v priebehu vývoja. Dôležité však je, že je tu možnosť voľby aproximačnej funkcie pre každý interval, vo forme roletového menu, takže v nerozvinutej podobe bude zaberať len jeden

riadok. Ďalšou časťou ktorá tu musí byť je zaškrtávacia voľba zvolenej aproximačnej metódy, kde vždy musí byť zaškrtnutá práve jedna možnosť. Posledným komponentom je samotný spúšťač výpočtu.

Textový výstup

V tejto oblasti bude vyobrazený analytický predpis výslednej funkcie vzniknutej aproximáciou zadaných bodov. Podobne ako hlavný grafický výstup bude aj textový rozdelený na tri intervaly, ako je naznačené na obrázku 6.1. Na základe takéhoto vyobrazenia je možné vypustiť z plochy grafu legendu a vyobrazené krivky v časti grafického výstupu programu budú popísané analytickým vzorcom práve v časti textového výstupu. Samozrejmosťou je vhodná voľba zvýrazňovacích prostriedkov na zobrazenie výsledných analytických predpisov, aby boli dostatočne odlíšené od ostatných textových komponent grafického užívateľského rozhrania.

6.3 Logický návrh

V tejto časti práce sa venujem popisu modulov programu. Ako už bolo spomenuté v úvode kapitoly, program bude vyvíjaný pomocou objektovo orientovaného prístupu, tieto moduly, resp. triedy sú následne popísané detailnejšie samostatne. Charakteristikou týchto modulov bude zároveň priblížený formát očakávaných vstupných dát, možnosť samostatného využitia modulov ako aj ich vzájomná interakcia.

Každý modul reprezentuje jednu triedu v ponímaní objektovo orientovaného paradigmatu. Jedná sa o nasledujúce moduly:

1. Hlavné okno – zodpovedá za zobrazenie grafického výstupu, má na starosti réžiu spúšťania činnosti ostatných modulov, preto bol aj zvolený za hlavný modul.
2. Konfigurátor – má na starosti načítanie a kontrolu správnosti vstupných dát a ich uloženie najvhodnejším spôsobom pre použitie zvyšnými modulmi.
3. Plotter grafu – do časti hlavného okna pre vykreslenie grafu zobrazí všetky komponenty grafu.
4. Aproximátor – tento modul používa ku aproximácií zadaných vstupných bodov genetický algoritmus.
5. Matematické funkcie – modul pre výpočet hodnôt bodov pre vykreslenie všetkých typov matematických funkcií.

Ťažiskom vytvorenia zadanej aplikácie nie je objektovo orientovaný návrh, ale správny návrh genetického algoritmu. Z toho dôvodu, ako aj preto, že jednotlivé moduly je plánované vytvoriť ako samostatné znovu použiteľné jednotky, nebudem v práci uvádzať triedny

diagram. Vzťahy medzi navrhnutými triedami sú priamočiare a dáta si vymieňajú pomocou parametrov. Takisto aj konkrétna podoba atribútov a metód tried vyplynie až v priebehu implementácie.

6.3.1 Hlavné okno

Tento modul bude hlavným riadiacim modulom, ktorý bude sledovať všetky akcie vyvolané užívateľom a na ich základe spúšťať činnosť zvyšných modulov. Okrem réžie ostatných tried bude mať na starosti aj vykreslenie komponent grafického užívateľského rozhrania. Počas implementácie bude vhodné zvážiť agregovanie niektorých z modulov, prípadne použitie modulu ako atribútu hlavného okna.

6.3.2 Konfigurátor

Načítanie vstupných dát a ich prevedenie do vhodných dátových štruktúr bude riešené v tomto module. Okrem načítania bude modul zvládať aj kontrolu vstupu a vyhodnocovanie chýb vzniknutých počas načítavania. To môžu byť chyby syntaktické, ako napríklad čísla v nesprávnom formáte, ako aj sémantické, teda zadané vstupné hodnoty mimo povolených hodnôt.

Vstupné dáta budú vo forme konfiguračného súboru. Tento súbor bude vytvorený užívateľom na základe šablóny konfiguračného súboru detailne popísanej v užívateľskej príručke, kde sú uvedené aj všetky položky konfiguračného súboru. Formát tohto súboru bude zhodný so zaužívaným INI formátom konfiguračných súborov. Príklad štruktúry INI súboru, postačujúci pre potreby navrhovanej aplikácie je uvedený vo výpise 6.1.

Výpis 6.1: Popis štruktúry konfiguračného súboru formátu INI

```
//komentár sa píše za znaky „//“, takto bude popísaná základná štruktúra INI

//súbor môže byť rozdelený na viacero sekcii

//označenie prvej sekcie
[sekcia_jeden]

//hodnota sa premennej priraďuje spôsobom kľúč = hodnota
premenna_x = 23.3
premennaY = true
premenna_z = „obsah“

//ďalšia sekcia
[sekcia_dva]

//v tejto sekcii je možné použiť názvy premenných z iných sekcií
```

```
premenna_x = 12
//prípustná je aj viacnásobná definícia premenných
premenna_x = 10
premenna_x = 17
```

Na základe možností INI konfiguračných súborov, bol navrhnutý formát vstupného konfiguračného súboru nasledovne. Sekcie aj hodnoty sú uvádzané v anglickom jazyku.

Výpis 6.2: Štruktúra konfiguračného súboru

```
//základná štruktúra vstupného konfiguračného súboru

//dáta pre prvý interval
[section_one]
//zoznam bodov určených na aproximáciu, vo formáte point = [X;Y]
point = [-5,0;-2.2]
point = [-4,0;-2.425]
point = [-3,0;-2.8]
function = 2

//dáta pre druhý interval
[section_two]
//zoznam bodov určených na aproximáciu, vo formáte point = [X;Y]
point = [1.0;8.2]
point = [1.5;9.5]
point = [2.8;11.3]

//dáta pre tretí interval
[section_three]
//zoznam bodov určených na aproximáciu, vo formáte point = [X;Y]
point = [7.6;15.9]
point = [8.2;16.3]
point = [10.0;19.2]

//všeobecné nastavenia netýkajúce sa konkrétnych intervalov
[general]
//x-ové súradnice intervalov
int1 = -10.0
int2 = 0.5
int3 = 7.0
int4 = 12.0
//rozsah zobrazenia na grafe
x_from = -12
x_to = 12
y_from = -10
y_to = 10
```

Vo výpise 6.2 sú uvedené povinné aj nepovinné parametre, medzi povinné patria body pre každý interval, a hodnoty, vymedzujúce hranice intervalov. Ostatné parametre sú nepovinné. Do konfiguračného súboru bude v priebehu implementácie pridaná možnosť nastavovať parametre genetického algoritmu. Z uvedenej šablóny vyplýva, že konfigurátor bude musieť počítať na vstupe s desatinnými číslami oddelenými ako desatinnou bodkou tak aj čiarkou.

6.3.3 Plotter grafu

Základnou funkciou tohto modulu je vykreslenie aproximovaných kriviek. Okrem toho bude schopný vykresliť zadané vstupné body a rozdelí plochu grafu na zadané intervaly. okamžite po načítaní a spracovaní konfiguračného súboru. Ďalej bude samozrejmosťou vykreslenie všetkých náležitostí grafu akými sú súradnicové osy *x* a *y*.

Tento modul bude riadený hlavným oknom. Jeho činnosť bude vyžadovaná vždy, keď bude potrebná manipulácia s obsahom grafu. Úlohy spojené s kontrolou definičného oboru, prípadne matematickej korektnosti dát určených na vykreslenie nie sú v kompetencii tohto modulu.

6.3.4 Aproximátor

Modul aproximátor bude slúžiť na aproximáciu zadaných výsledkov meraní na nejakú zvolenú matematickú funkciu. Tento modul bude najrobustnejší a na svoju činnosť bude používať genetické algoritmy. Návrhu modulu aproximátor sa venujem v celej ďalšej podkapitole 6.4.

6.3.5 Matematické funkcie

Úlohou tohto modulu je združovať všetky triedy, ktoré vracajú hodnoty aproximovaných matematických funkcií. Jeho funkčnosť bude využívaná pri vykresľovaní kriviek do výsledného grafu. Implementované budú nasledujúce funkcie, vedľa názvu funkcie je aj jej analytické vyjadrenie:

- lineárna funkcia: $y = ax + b$,
- kvadratická funkcia: $y = ax^2 + bx + c$,
- hyperbolická funkcia (1): $y = \frac{a}{x} + b$,
- hyperbolická funkcia (2): $y = \frac{a}{x^2} + bx + c$,
- polynomiálna funkcia tretieho stupňa: $y = ax^3 + bx^2 + cx + d$,

- polynomiálna funkcia štvrtého stupňa: $y = ax^4 + bx^3 + cx^2 + dx + e$,
- exponenciálna funkcia (1): $y = ae^{-x} + b$,
- exponenciálna funkcia (2): $y = ae^{b(-x)} + c$,
- logaritmická funkcia: $y = a \ln x + b$,
- logaritmus sínusu: $y = a \ln(2 + \sin x) + b$,
- sínusoida: $y = a \sin(x + b) + c$.

Daný zoznam funkcií, ktorými bude možné aproximovať zadané body nemusí byť konečné, preto treba pri implementácií celého programu dbať na vysokú rozšíriteľnosť kódu.

6.4 Návrh genetického algoritmu

Jadro celej aplikácie bude tvoriť genetický algoritmus. Preto je pre správnu funkcionalitu programu kľúčovou časťou návrh genetického algoritmu. Táto časť sa preto zaoberá a mapuje postupné kroky návrhu genetického algoritmu, ktorý by dokázal aproximovať zadané body zvolenou funkciou prostredníctvom metódy najmenších štvorcov, alebo metódy najmenších vzdialeností. Naviac je nutné, aby tento algoritmus bral v úvahu aproximáciu v rámci troch intervalov s následnou spojitosťou výsledných funkcií na zadaných intervaloch.

Riešenie spojitosti na intervaloch bude popísané v záverečnej časti tejto podkapitoly. Postup práce som si rozložil tak, že najskôr prebehne návrh genetického algoritmu na samotnú aproximáciu a následne sa tento upraví tým spôsobom, aby aproximácia prebehla v každom intervale s dodržaním spojitosti funkcií. Preto sa popis genetického algoritmu sústredí len na samotnú aproximáciu.

Pseudokód genetického algoritmu je zobrazený na výpise 6.3 [21]. Implementácia, ktorú tento výpis popisuje, zároveň definuje jednotlivé kroky návrhu a procedúry, ktoré treba definovať, respektíve vhodne zvoliť. Tieto kroky boli načrtnuté v kapitole 3.4.

Výpis 6.3: Pseudokód genetického algoritmu

```
BEGIN
  INICIALIZUJ populáciu náhodnými kandidátskymi riešeniami;
  OHODNOŤ každého jedinca;
  REPEAT UNTIL (UKONČOVACIA PODMIENKA nie je splnená) DO
    VYBER rodičov;
    REKOMBINUJ páry rodičov;
    MUTUJ nového potomka;
    OHODNOŤ vzniknutých jedincov;
    VYBER individual do ďalšej generácie;
  OD
END
```

6.4.1 Voľba reprezentácie problému

Prvá otázka, ktorú je nutné vyriešiť pri návrhu genetického algoritmu je spôsob reprezentácie problému. Táto voľba má silné dopady na ďalší návrh algoritmu a pri nevhodne zvolenej reprezentácii môže mať algoritmus vysoké časové nároky na dosiahnutie riešenia, prípadne sa nemusí ku suboptímálnemu riešeniu dopracovať vôbec. V tomto kroku je potrebné zvoliť jednak podobu jedincov, teda akú formu budú mať konkrétne kandidátske riešenia, ako aj ich číselnú reprezentáciu.

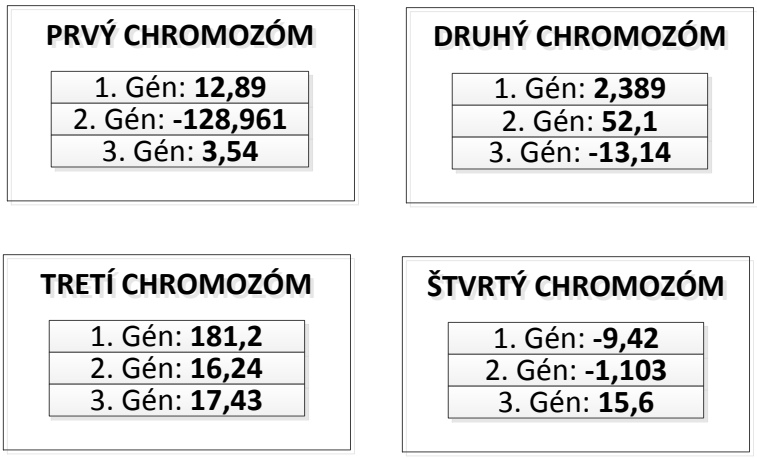
Vychádzajúc z analýzy požiadaviek na program je cieľom genetického algoritmu poskytnúť určitý počet reálnych koeficientov aproximovaných matematických funkcií. Ich počet je závislý od typu funkcie, kde napríklad pre kvadratickú funkciu sú to tri koeficienty ($y = \boldsymbol{a}x^2 + \boldsymbol{b}x + \boldsymbol{c}$) a pre polynomiálnu funkciu tretieho stupňa budú hľadané štyri reálne koeficienty $y = \boldsymbol{a}x^3 + \boldsymbol{b}x^2 + \boldsymbol{c}x + \boldsymbol{d}$. Tým pádom je výhodné, pokiaľ bude jedinec, reprezentujúci možné riešenie úlohy, reprezentovaný štruktúrou reálnych čísel, ktoré zodpovedajú koeficientom aproximovanej funkcie. Táto štruktúra môže mať podobu poľa, zoznamu, fronty, atp. Vzhľadom na najnižšie pamäťové nároky bude zvolená reprezentácia jedincov vo forme poľa reálnych hodnôt. Populáciu budú tvoriť teda chromozómy, implementované ako polia obsahujúce reálne čísla, a tieto reálne čísla budú v terminológii genetických algoritmov zastupovať gény.

Pre voľbu číselnej reprezentácie génov sa naskytujú dve možnosti:

- Binárna reprezentácia s implementáciou niektorej z metód prevodu do desiatkovej sústavy, tu by však bol problém s vyhradením dostatočného počtu bitov pre desatinnú časť na dosiahnutie vysokej presnosti výpočtu ako aj možná vyššia časová náročnosť algoritmu z dôvodu vysokého počtu prevodov medzi binárnou a desiatkovou reprezentáciou, táto možnosť teda nie je vhodná a nebude ďalej uvažovaná.

- Ďaleko vhodnejšia a priamočiarejšia ako prvá možnosť je priama reprezentácia génov pomocou reálnych čísiel. Pozitívom tejto voľby je odstránenie potreby kódovania a dekódovania hodnoty génu ako aj počet desatinných miest a presnosť je daná len obmedzeniami použitého dátového typu.

Náčrt podoby populácie so štyrmi jedincami, ktoré sú kandidátskymi riešeniami pre aproximáciu kvadratickej funkcie je na obrázku 6.4.



Obrázok 6.4:
Príklad populácie o veľkosti 4 pre matematickú funkciu s tromi parametrami

Na obrázku 6.4 je zobrazený chromozóm ako abstraktná štruktúra obsahujúca tri položky, gény, ktoré budú v programe nositeľmi len reálnej hodnoty. Gén tu teda nie je nositeľom len jednej číslice, ale celého čísla.

6.4.2 Pôvodná inicializácia populácie

Inicializácia populácie prebehne tak, ako to býva v prípade využívania genetických algoritmov zvykom, a teda pridelením náhodných hodnôt jednotlivým génom. Najväčšou slabinou týchto algoritmov je nutnosť vymedzenia určitých hraníc prehľadávacieho priestoru. V tomto prípade je to zadanie rozsahu počiatočných hodnôt génov. Teda určenie intervalu v ktorom sa budú náhodné reálne čísla pre počiatočné hodnoty génov generovať. Pri zvolení vysokého rozmedzia hrozí riziko príliš dlhej konvergencie riešenia ku optimálnemu riešeniu, čím sa môže neprijateľne predĺžiť doba behu programu. Na druhej strane veľmi krátky interval podstatne limituje veľkosť prehľadávaného priestoru, a je vysoko pravdepodobné, že nájdené riešenie bude veľmi vzdialené optimálnemu.

Voľbu najvhodnejšieho horného a dolného obmedzenia inicializácie nechávam čiastočne na experimentovanie pri implementácií. Avšak podľa literatúry [21] genetické algoritmy dokážu rýchlo dospieť ku riešeniu blízkemu optimu za relatívne krátku dobu aj v prípade vysokého rozptylu hodnôt počiatočnej generácie. Takisto sa tu pri veľkom rozsahu počiatočného intervalu dá v záujme nájdenia čo najoptimálnejšieho riešenia operovať aj

s veľkosťou populácie, prípadne ukončovacou podmienkou pre algoritmus. Preto volím ako prvotný interval vymedzujúci hodnoty parametrov hľadaných aproximovaných funkcií na <-2000, +2000>. Tento nedostatok genetického algoritmu bude čiastočne odstránený poskytnutím možnosti nastaviť interval hľadania riešenia užívateľovi, ktoré bude umožnené z dôvodu predpokladu, že s programom bude pracovať vedecký pracovník. Ten by mal byť schopný daný rozptyl odhadnúť, pokiaľ žiadnu hodnotu nezadá, bude sa počítať s implicitne nastavenou hodnotou.

6.4.3 Spôsob ohodnocovania jedincov populácie

Po určení akým spôsobom bude problém zakódovaný do populácie, jedincov a ich génov zároveň s ich počiatočnou inicializáciou, je potrebné zvoliť spôsob akým budú jedinci, zastupujúci riešenie ohodnocovaní tak, aby ich vhodnosť zastávať čo najlepšie riešenie bola medzi sebou porovnateľná a na základe tohto ohodnotenia aby mali títo jedinci zodpovedajúcu šancu na tvorbe potomkov, teda vytvorenie novej populácie. Ide o určenie fitness hodnoty jedincov.

Keďže jedincom sa rozumie pole parametrov hľadanej aproximačnej funkcie, bude jeho fitness určovaná na základe toho, ako dobre dokáže funkcia ktorú jedinec reprezentuje aproximovať vstupné body zadané užívateľom. V tomto procese zohráva najväčšiu rolu metóda najmenších štvorcov, resp. najmenších vzdialeností. Práve suma druhej mocniny *y*-ovej vzdialenosti všetkých zadaných bodov od funkcie ktorej konkrétny predpis nesie jedinec bude rozhodujúcim elementom pre vyhodnotenie fitness tohto jedinca. Keďže je zaužívané pravidlo, že čím vyššia je fitness tým je jedinec vhodnejší, v tomto konkrétnom prípade bude nutné sumu druhej mocniny vzdialeností všetkých bodov prevrátiť. Teda počítať sa bude s obrátenou hodnotou tejto sumy. Pokiaľ bude aproximované pomocou metódy najmenších vzdialeností, nebude sa rátať s druhou mocninou *y*-ovej vzdialenosti ale s absolútnou hodnotou tejto vzdialenosti. Fitness *i*-tého jedinca v populácií bude pri použití metódy najmenších štvorcov a aproximácií pomocou funkcie *f* počítaná nasledujúcim vzťahom:

$$fitness_i = \frac{100}{\sum_{j=1}^{N} \left(y_j - f\left(x_j\right) \right)^2}$$

(6.1)

Vo vzťahu 6.1 je *N* počet zadaných vstupných bodov, *i* je poradie jedinca v populácii, *j* zodpovedá poradovému číslu bodu zadaného užívateľom kde *y* a *x* sú jeho súradnice a *f* je aproximovaná funkcia. To isté platí aj pri použití metódy najmenších vzdialeností, tu by bol vzťah na vypočítanie fitness pre *i*-tého jedinca takýto:

$$fitness_i = \frac{100}{\sum_{j=1}^{N} |y_j - f(x_j)|}$$

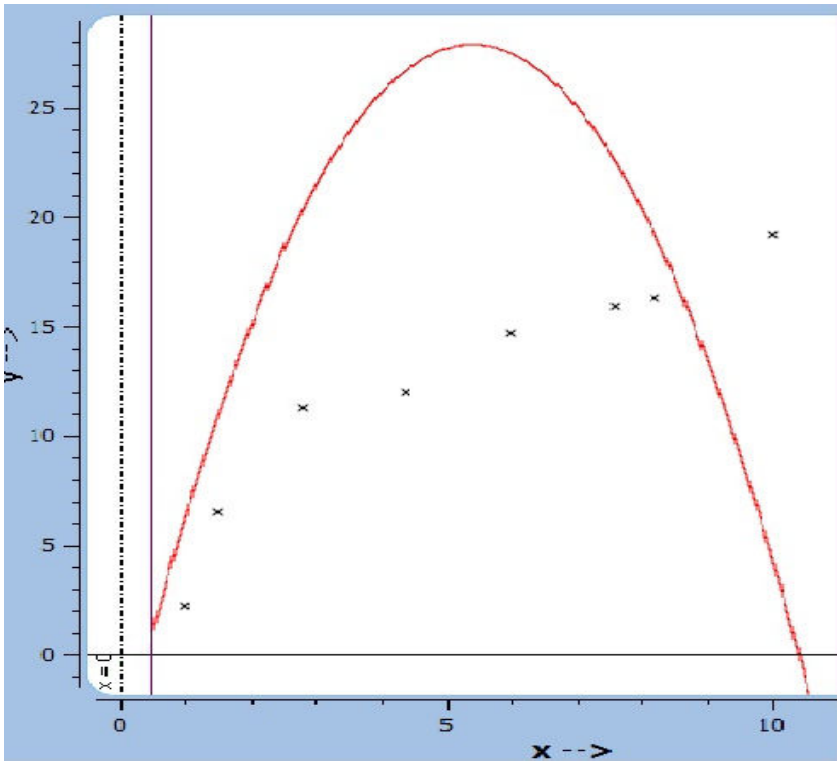
(6.2)

Vzhľadom na vysoký počet desatinných miest reálnych dátových typov a ich presnosť, nie je potrebné zvyšovať hodnotu čitateľa, aj keď hodnota menovateľa bude minimálne v počiatočných generáciách riešenia vysoké číslo. Pre ilustratívnosť je uvedený nasledujúci vzťah výpočtu fitness hodnoty jedinca pre aproximáciu logaritmickou funkciou s použitím metódy najmenších štvorcov, kde parametre *a* a *b* sú tvoria hľadané riešenie a konkrétny jedinec ich v programe bude mať definované.

$$fitness_i = \frac{100}{\sum_{j=1}^{N} (y_j - a \ln(x_j) + b)^2}$$

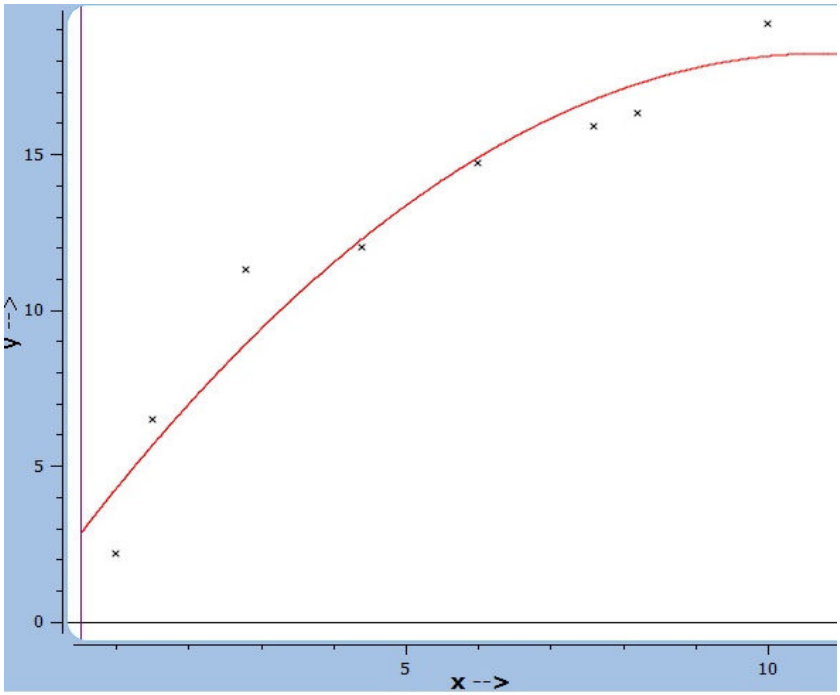
(6.3)

Na obrázku 6.5 je červenou farbou zobrazená funkcia, zodpovedajúca jedincovi, ktorého koeficienty určujú predpis kvadratickej funkcie, pomocou ktorej sú aproximované zadané body zobrazené symbolom krížik.



Obrázok 6.5:
Zobrazenie fenotypu pre krivku reprezentovanú jedincom s nízkou fitness

V terminológií genetických algoritmov predstavuje daná krivka na obrázku 6.5 fenotyp genotypu *a = -1.1122*, *b = 12.0394*, *c = -4.7100*, kde štruktúra týchto troch hodnôt tvorí jedinca populácie, respektíve jeden chromozóm. Analytické vyjadrenie tejto krivky je $y = -1.1122x^2 + 12.0394x - 4.7100$. Fitness hodnota tohto jedinca je oproti jedincovi vyjadrujúcim zhodnú skutočnosť s inou krivkou na obrázku 6.6 nízka. Naopak fitness jedinca, ktorého fenotyp je červená krivka na obrázku 6.6, je veľmi vysoká.



Obrázok 6.6:
Zobrazenie fenotypu pre krivku reprezentovanú jedincom s vysokou fitness

Jednotlivé hodnoty génov chromozómu, ktorého fenotyp je zobrazený na obrázku 6.6 sú *a = -0.1458*, *b = 3.1466*, *c = 1.2498*. Analytickým vyjadrením tejto krivky je $y = -0.1458x^2 + 3.1466x - 1.2498$. Rozdiel vo vhodnosti reprezentovať čo najoptimálnejšie riešenie je medzi týmito dvoma zobrazenými prípadmi veľmi dobre viditeľný. Body na obrázku 6.6 sú aproximované krivkou oveľa lepšie ako tie na obrázku 6.5. Teda suma druhých mocnín ich vzdialeností na *y*-ovej súradnici je značne nižšia ako táto suma pre prípad na obrázku 6.5. Keďže fitness bude počítaná ako prevrátená hodnota sumy vzdialeností, je zákonite fitness hodnota jedinca z obrázka 6.6 vyššia.

6.4.4 Selekcia párov rodičovskej populácie

Po ohodnotení každého jedinca populácie fitness hodnotou je potrebné určiť, akým spôsobom budú z tejto populácie vyberané jedince do procesu tvorby potomkov, teda do procesu kríženia. Ako rodičia budú zvolení jedinci s najvyššou fitness.

V teoretickej časti, v kapitole 3.3.3 boli popísané najpoužívanejšie metódy selekcie s ich výhodami aj nevýhodami. Pre zvolenie konkrétneho prístupu ku selekcii rodičov som previedol predčasný prieskum nástrojov určených na implementáciu genetických algoritmov. Z neho vyplynulo, že vo všetkých preskúmavaných nástrojoch je proporciálny výber na základe fitness hodnoty, tzv. selekcia ruletovým kolesom, implicitnou metódou selekcie rodičovských párov. Takisto sa aj v mnohých aplikáciách spísaných v kapitole číslo 2 uvádza ruletový výber ako používaný spôsob výberu rodičov. Ďalším dôležitým zistením bolo, že zmena selekčného algoritmu býva často veľmi jednoduchý úkon prepísania konštanty vo volaní funkcie. Preto sa nebudem ďalej zaoberať analýzou možností selekcie rodičovských párov a ako východzia bude proporciálna selekcia na základe fitness hodnoty, ruletová selekcia. Táto sa bude dať na základe experimentovania s finálnym programom kedykoľvek zmeniť. Uvedené závery platia však len v tom prípade, že sa rozhodnem pre implementáciu využiť nástroj tretej strany a neprikloním sa k implementácii vlastného nástroja. V tomto prípade by ale implementácia viacerých metód selekcie rodičov taktiež nemal byť problém.

Rozhodnutie o finálnej podobe algoritmu selekcie rodičov podobne ako intervaly prvotnej inicializácie ponechávam na experimentálne určenie. Takisto je možné nechať voľbu selekcie rodičovských párov na užívateľa formou parametru konfiguračného súboru.

6.4.5 Kríženie rodičovských párov

Na vznik nových potomkov, s potenciálne lepšou fitness hodnotou ako ich rodičia je potrebné rozhodnúť o spôsobe ich vzniku.

Voľba reprezentácie jedincov formou poľa reálnych čísel zúžila spôsob kríženia rodičovských chromozómov na aritmetické kríženie (viď 3.3.4) podľa vzťahu 3.4. Z dôvodu, že jednotlivé gény v rámci chromozómu v mnou zvolenej reprezentácii problému medzi sebou nesúvisia, musí dochádzať ku kríženiu génov dvoch chromozómov s rovnakým indexom. Teda vždy je to parameter *a* prvého chromozómu s parametrom *a* druhého chromozómu, následne parameter *b* prvého chromozómu s parametrom *b* druhého chromozómu. Pre prípad aproximácie pomocou logaritmickej funkcie $y = a \ln x + b$, by kríženie rodičovských chromozómov za účelom výpočtu hodnôt génov novovzniknutých potomkov prebehlo podľa nasledujúceho vzťahu:

$$\begin{aligned}
prvý_potomok[0] &= \alpha * prvý_rodič[0] + (1-\alpha) druhý_rodič[0] \\
prvý_potomok[1] &= \alpha * prvý_rodič[1] + (1-\alpha) druhý_rodič[1] \\
druhý_potomok[0] &= (1-\alpha) prvý_rodič[0] + \alpha * druhý_rodič[0] \\
druhý_potomok[1] &= (1-\alpha) prvý_rodič[1] + \alpha * druhý_rodič[1]
\end{aligned} \tag{6.4}$$

V prípade zachytenom vzťahom 6.4 je chromozóm reprezentovaný jednorozmerným poľom o veľkosti dvoch prvkov, prvý parameter *a* má index *0* a druhý hľadaný parameter *b* má index *1, α* je náhodne generované reálne číslo, pre ktoré platí $0 \leq \alpha \leq 1$. Vzťah 6.4 platí pre kríženie pri akejkoľvek zvolenej aproximačnej funkcii, meniť sa bude len počet iterácií výpočtu v závislosti od počtu génov chromozómu, resp. počtu parametrov aproximačnej funkcie.

Po definovaní spôsobu kríženia zostáva určiť pravdepodobnosť, s akou bude medzi rodičmi ku kríženiu dochádzať. Z dôvodu rýchlejšej konvergencie ku optimálnemu riešeniu stanovujem túto hodnotu na 90%. To znamená, že v deväťdesiatich percentách prípadov dôjde po selekcií dvoch rodičovských párov ku ich kríženiu a vzniku dvoch nových potomkov. Túto hodnotu bude samozrejme možné na základe empirických experimentov s programom meniť.

6.4.6 Mutácia

Ako už bolo uvedené, z dôvodu vnesenia nového impulzu, istej zmeny, do vývoja populácie je dôležité definovať operáciu mutácie a pravdepodobnosť jej výskytu.

Tak ako aj pre kríženie sú mutačné operátory v tejto úlohe obmedzené na aritmetické operácie. V programe budú jednotlivé gény mutované na základe nasledujúceho vzťahu:

$$zmutovaný\_chromozóm[i] = pôvodný\_chromozóm[i] * \alpha$$

(6.5)

Mutácia nesmie zmeniť jedinca do takej miery, aby extrémne degradovala jeho schopnosť zastávať riešenie, preto bude reálny parameter α vo vzťahu 6.5 náhodne generovaný v rozmedzí <0,9 ; 1.1>. Jedná sa o zmenu hodnoty génu maximálne o desať percent.

Pravdepodobnosť, s ktorou mutácia nastane bude primárne nastavená na 1%. S takouto pravdepodobnosťou však bude dochádzať ku mutáciu na jednotlivých génoch, nie na každom géne chromozómu.

6.4.7 Spôsob obnovy populácie

Po prebehnutí doteraz popísaných procesov bude vytvorená populácia potomkov, a pôvodná populácia, z ktorej boli niektorí jedinci zvolení ako rodičia. Následne prebehne ohodnotenie potomkov rovnakým spôsobom aký je popísaný v kapitole 6.4.3, aby mal každý potomok pridelenú fitness hodnotu. Nastáva krok výberu jedincov do novej populácie.

Tento proces bude zhodný so selekciou rodičovských párov, teda z oboch vyššie zmienených populácií bude ruletovou selekciou vybraných vopred určený počet (zodpovedajúci veľkosti populácie) jedincov, ktorý vytvoria novú generáciu riešení a v ďalšej iterácii behu

genetického algoritmu budú tvoriť rodičovskú populáciu a na základe ich fitness hodnoty budú súťažiť o účasť v tvorbe novej generácie potomkov. To znamená, že jedinci s najvyššou fitness budú mať najväčšiu pravdepodobnosť zahrnutia do novej generácie.

Okrem spôsobu tejto selekcie si genetické algoritmy vyžadujú stanoviť veľkosť populácie, ktorá bude nahradená novými potomkami. Z naštudovaných zdrojov túto veľkosť stanovujem na 50% celej populácie.

Výber rodičov aj jedincov ktorí prežijú je stále založený na pravdepodobnostných hodnotách, a tak sa môže s nízkym percentom pravdepodobnosti stať, že jedinec s najlepšou fitness nebude zvolený do tvorby ďalšej populácie, preto bude v navrhovanom programe zapnutý proces elitizmu, ktorý zabezpečí prejdenie najlepšieho jedinca populácie do novej generácie.

6.4.8 Ukončenie algoritmu

Pre ukončenie genetického algoritmu, a teda skončenie výpočtu existujú dva používané prístupy. Prvým je ukončenie pri istej, stanovenej miere konvergencie ku jednému optimu. Teda nové generácie jedincov neprinášajú nové riešenia a algoritmus dospel ku najoptimálnejšiemu riešeniu. Zrejmým obmedzením takéhoto spôsobu ukončovania algoritmu je teoreticky potenciálny nekonečný beh algoritmu, kedy nebudú jedince konvergovať dostatočne. Z toho dôvodu volím pre tento program spôsob ukončenia na základe počtu iterácii. Algoritmus sa po určitom počte populácií zastaví, a nositeľom najoptimálnejšieho riešenia bude jedinec s najvyššou fitness v poslednej generácii. Finálny počet generácií bude empiricky stanovený v priebehu testovania programu, vopred ho určujem na sto generácií.

6.4.9 Spojitosť aproximovaných funkcií na intervaloch

Zo špecifikácie zadania vyplýva, že užívateľ zadá tri intervaly, v ktorých prebehne aproximácia, zároveň volí tri aproximačné funkcie, pre každý interval jednu. Ďalej zo zadania vyplýva nutnosť spojitosti aproximovaných funkcií na celej ploche vymedzenej zadanými intervalmi. Doteraz popisovaný postup zachytával aproximáciu funkcie len v jednom intervale, kde by zákonite na hranici intervalov dochádzalo ku diskrétnym skokom medzi funkciami. Tento jav je silne nežiaduci a pokiaľ by sa aplikácia takto chovala, tak by nespĺňala zadanie. Miernou modifikáciou a doplnením popísaného postupu je možné upraviť chovanie aproximácie tak, aby bola spojitosť výsledných funkcií dodržaná.

Od užívateľa budú zadané štyri hodnoty, ktoré rozdelia plochu riešenia na tri intervaly. Pre zabezpečenie spojitosti funkcií v týchto troch intervaloch vymedzených užívateľom som definoval nasledujúci postup:

1. Výpočet aproximovanej funkcie pre prostredný interval prebehne nemodifikovaným postupom popísaným v predchádzajúcich podkapitolách.

2. Po vypočítaní aproximačnej funkcie prostredného intervalu sa získajú priesečníky tejto funkcie s hranicami prostredného intervalu, *x*-ová súradnica týchto priesečníkov bodu je daná užívateľom ako hranica vymedzujúca interval, *y*-ová sa dopočíta dosadením *x*-ovej hodnoty do analytického vyjadrenia priamky získanej genetickým algoritmom. Tieto body sú bodmi nutnej spojitosti aproximovaných priamok, teda tu sa budú aproximované priamky prvého a tretieho intervalu stretávať.

3. V ďalšom postupe bude popísanie získania aproximačnej funkcie pre prvý interval, pre tretí interval je tento výpočet zhodný. Funkcie pre prvý a tretí interval sú závislé od priesečníkov vypočítaných v bode 2. Je známe, akým bodom musí výsledná krivka prvého intervalu prechádzať, vďaka tomuto je možné vypočítať funkčnú závislosť každého nezávislého parametru matematických funkcií. Nasledujúci vzťah demonštruje odvodenie výpočtu nezávislého parametru *c* pre kvadratickú funkciu. Vo vzťahu 6.6 *y* zodpovedá *y*-ovej súradnici priesečníka s hranicou medzi prvým a druhým intervalom z bodu 2, a *x* je potom *x*-ová hodnota tohto priesečníka.

$$c = y - (ax^2 + b) \tag{6.6}$$

4. Z uvedeného vyplýva, že modifikácia genetického algoritmu nastane skrátením dĺžky chromozómu, teda počtu jeho génov o jeden. To nastane z toho dôvodu, že posledný, nezávislý parameter každej funkcie bude vždy odvodený od zvyšných parametrov. Týmto sa zaručí spojitosť na celej ploche hľadania riešenia. Pre kvadratickú funkciu bude potom chromozóm obsahovať dva gény, dve reálne hodnoty, a to parameter *a* a parameter *b,* pričom parameter *c* bude vždy odvodený podľa vzťahu 6.6.

5. Rovnaký postup, avšak s priesečníkom medzi na hranici druhého a tretieho intervalu bude aplikovaný na získanie aproximovanej funkcie v treťom intervale.

Popísaním dodržania spojitosti aproximovaných funkcií je dokončená fáza analýzy a návrhu riešenia. Ďalším krokom je samotná implementácia, ktorej popisu sa venuje celá nasledujúca kapitola.

7 Vývoj programu

Pred samotnou implementáciou bolo potrebné previesť výber programovacieho jazyka, vývojového prostredia, nástrojov a knižníc tretích strán, ktoré boli na vývoj použité. Tomuto sa venuje prvá časť kapitoly. Z implementačnej časti je tu aj s výpiskami zdrojových kódov z dôvodu zamerania práce uvedená len časť venujúca sa vývoju genetického algoritmu, kompletné zdrojové kódy sú umiestnené v prílohe. Ďalej sa v tejto kapitole venujem popisu experimentovania s parametrami implementovaného genetického algoritmu, ktoré je zobrazené formou grafov a sprievodného textu. Na konci kapitoly je uvedený extrakt z užívateľskej príručky. Kompletný návod na použitie sa nachádza v prílohe práce.

7.1 Programovací jazyk a vývojové prostredie

Vo fáze analýzy bolo rozhodnuté, že program bude vyvíjaný formou spustiteľnej aplikácie. To zúžilo možnosti výberu programovacieho jazyka, zároveň boli zamietnuté možnosti vývoja formou skriptu s následnou konverziou na spustiteľnú aplikáciu. Z možných implementačných jazykov zostali traja kandidáti, ktorých som vybral na základe mojej znalosti vývoja v týchto jazykoch:

- jazyk C,
- jazyk C++,
- jazyk Java.

Z dôvodu rozhodnutia vývoja pomocou objektovo orientovaného prístupu som musel z možností odstrániť jazyk C, ktorý neumožňuje použitie prostriedkov tohto prístupu. Výber sa tak zúžil na jazyk C++, ktorý je objektovou nadstavbou jazyka C, a Javu. Výhodou Javy oproti C++ je prenositeľnosť kódu, ktorú však zadanie nevyžaduje. Vývojové prostredia sú pre oba jazyky dostupné vo vyššom počte a ich kvalita je rokmi overená, takže z tohto hľadiska nebol výhodnejší ani jeden z jazykov. To isté platí aj pre veľmi rozšírenú komunitu programátorov v oboch jazykoch, s čím súvisí široká podpora pre ktorýkoľvek z nich. Hlavne z dôvodu mojej hlbokej znalosti jazyka C++, širokými skúsenosťami s týmto jazykom a obzvlášť tak s jazykom C som sa rozhodol priklоniť k vývoju pomocou jazyka C++.

S výberom implementačného jazyka úzko súvisí voľba vhodného vývojového prostredia, ktoré dokáže ušetriť mnoho času a byť vývojárovi veľmi silným pomocným prostriedkom. Keďže sa jedná o vývoj aplikácie s grafickým užívateľským rozhraním, bolo potrebné vyhľadať najvhodnejších kandidátov, ktorý by v najlepšom prípade poskytovali možnosť interaktívnej, drag & drop, formy tvorby grafického užívateľského rozhrania (GUI) a zároveň podporovali vývoj na operačnom systéme Windows 7.

Po prieskume v oblasti voľne dostupných GUI nástrojov pre C++ som dospel ku dvom vhodným kandidátom:

- wxWidgets,
- Qt.

wxWidgets toolkit poskytuje rozsiahle API na tvorbu grafických aplikácií, podporujúci viacero platforiem. S nástrojom Qt sú si veľmi podobné v robustnosti a rozsiahlosti, preto som sa zameral na porovnanie ich odlišností, z ktorého vyšlo Qt ako lepší nástroj pre použitie na implementáciu danej úlohy. Vzájomné odlišnosti sú uvedené v nasledujúcom zozname.

- Najväčším nedostatkom wxWidgets je veľmi slabá dokumentácia, ktorá naviac nie je centrálne umiestnená, ale mnoho vecí je potreba hľadať na rôznych miestach. Oproti tomu dokumentácia Qt [34] je centralizovaná, prehľadná a vyčerpávajúca.
- Zložitá inštalácia a samotná kompilácia wxWidgets na systéme Windows, kde inštalácia Qt je jednoduchá a priamočiara.
- Jednoduchá manipulácia s makefile súbormi pri použití Qt na rozdiel od komplikovaného editovania týchto súborov pod wxWidgets.
- Qt poskytuje jednoduchú podporu separácie programovania GUI od zvyšného programového kódu, takáto možnosť vo wxWidgets nie je.
- Qt ponúka prehľadné vývojové prostredie QtCreator s dizajnérskym pluginom pre jednoduchú tvorbu GUI.
- Systém slotov a signálov podporovaný Qt.

Na základe prevažujúcich pozitív nástroja Qt, som sa rozhodol použiť pre vývoj programu framework Qt.

7.1.1 Framework Qt

Pre účely tejto práce nie je potrebný vyčerpávajúci popis tohto frameworku, preto tu uvediem len jeho základnú charakteristiku. V tomto popise vychádzam z informácií uvedených v [35], čo je aj dobrým zdrojom dodatočných informácií pre záujemcov.

Základy Qt frameworku siahajú do roku 1995, kedy bola publikovaná jeho prvá verzia. Od roku 1998 sú nové aktualizácie a nové verzie vydávané pod hlavičkou firmy Trolltech. Na vývoji frameworku sa však podieľa celá komunita programátorov a firmy ako Nokia a Trolltech.

Tento framework používa vlastné renderovacie jadro, ktorým sa snaží emulovať prostredie systému na ktorom beží, týmto uľahčuje prenositeľnosť medzi rôznymi verziami Windows. Ďalej plne podporuje štandardnú knižnicu C++, takže sa pod ním dajú vyvíjať aj konzolové aplikácie, bez nutnosti použitia akýchkoľvek tried frameworku. Okrem štandardnej knižnice ponúka svoje vlastné moduly ako napríklad modul na sieťové programovanie, modul pre integráciu s SQL prostredím, pre prácu s OpenGL knižnicou a tvorbou 3D grafiky, vlastný modul pre prácu s XML a mnoho ďalších.

Qt predstavil prístup programovania grafického rozhrania pomocou signálov a slotov a doteraz ho vo svojom frameworku umožňuje používať. Je to spôsob komunikácie medzi objektmi, kedy jeden objekt vyšle pri zmene svojho stavu signál, pokiaľ je pre okolité objekty táto zmena dôležitá tak na ňu reagujú. Sloty sú obyčajné C++ metódy objektov, ktoré sú zavolané pokiaľ je vyslaný signál, na ktorý reagujú. Definícia slotov a signálov na ktoré reagujú je plne v réžii programátora.

QtCreator je vývojové prostredie, určené pre programovanie v C++ a Qt. Toto prostredie poskytuje klasické možnosti iných vývojových prostredí, akým je zvýrazňovanie syntaxe, automatické odsádzanie, výkonný nástroj na krokovanie kódu atp. Jeho výraznou pozitívnou črtou je previazanie s pluginom Qt designer, ktorý umožňuje interaktívnu tvorbu GUI, definíciu signálov a slotov a inteligentné umiestnenie grafických komponent veľmi jednoduchým grafickým spôsobom.

7.2 Použité technológie

Zo záverov vyplývajúcich z logického návrhu, sa program bude skladať z viacerých modulov. Pred implementáciou je treba zvoliť, ktorý z týchto modulov bude plne samostatne implementovaný, respektíve aké konkrétne riešenia tretích strán najlepšie spĺňajú požiadavky na funkčnosť. Pre veľkú rozsiahlosť v texte neuvádzam alternatívne nástroje a dôvody zvolenia, resp. nezvolenia, ale popísané sú len použité knižnice.

7.2.1 Parsovanie konfiguračného súboru

Modul pre prácu s konfiguračným súborom využíva na získanie hodnôt zo súboru parsovaciu knižnicu pre konfigurátory v INI formáte s názvom SimpleIni. Táto knižnica plne spĺňa požiadavky sumarizované v kapitole 6.3.2. U veľkej väčšiny iných knižníc určených na načítanie hodnôt z INI súborov absentovala kľúčová možnosť mať v súbore zadaný viacnásobný kľúč. V tomto prípade sa jedná o kľúč *point*, definujúci dvojicu súradníc vstupných bodov.

SimpleIni je multiplatformová knižnica distribuovaná formou jedného súboru na úrovni zdrojového kódu. Tento súbor bolo následne potrebné pripojiť k projektu a skompilovať. SimpleIni poskytuje nasledujúcu funkčnosť po načítaní súboru v INI formáte:

- získanie celého obsahu jednej sekcie, prípadne získanie všetkých kľúčov v rámci jednej sekcie,
- získanie hodnoty kľúča v konkrétnej sekcii, hodnota môže byť typu *bool, double, string, long,*
- získanie hodnôt viacnásobného kľúča pre konkrétnu sekciu a ich uloženie do vektora,
- pridanie novej sekcie alebo nového kľúča,
- modifikovanie hodnoty konkrétneho kľúča,
- zmazanie celej sekcie alebo kľúča.

SimpleIni je implementovaná v jazyku C++, preto v nej nebol problém previesť drobné úpravy kódu, hlavne podporu desatinných bodiek aj čiarok pre reálne čísla. Ukážky zdrojových kódov ako aj detailná dokumentácia je uvedená v [36].

7.2.2 Plotter grafu

Hlavným výstupom aplikácie bude zobrazenie bodov a aproximačných priamok formou grafu. Z toho dôvodu je potrebné zvoliť na vykreslenie grafu taký nástroj, ktorý ponúka dostatočne rozsiahlu podporu pre vykresľovanie grafu.

Najlepšia knižnica pre vykresľovanie grafov, ktorá je voľne dostupná a je vyvinutá špeciálne pre framework Qt je knižnica s názvom Qwt. Jej inštalácia prebehne formou kompilácie do podoby dynamickej knižnice a pri používaní Qwt programom je potrebné túto dynamickú knižnicu zahrnúť do prekladu. Qwt je implementované v jazyku C++ avšak integrovaná môže byť aj do projektov v jazyku python a ruby.

Qwt podporuje základné operácie pre vykreslenie grafu akými sú popis súradnicových osí, vloženie kriviek do grafu s daným analytickým vyjadrením, vloženie legendy a popiskov do grafu, možnosť priblíženia a oddialenia grafu a mnoho ďalšej základnej funkcionality. Z pokročilejšej funkcionality je to možnosť vykreslenia interaktívnych grafov s manipulovateľnými elementmi, použitie rôznych ovládacích prvkov na zmenu parametrov grafu, podpora vykresľovania skutočností (zaťaženie procesoru, využitie operačnej pamäte) v reálnom čase. Rozsiahlosť použitia a vysoké množstvo funkcionality však neznamená zvýšenú zložitosť programovania.

Qwt poskytuje rozsiahlu užívateľskú dokumentáciu [37], kde sú popísané všetky moduly tejto knižnice, takisto aj programátorská komunita je rozšírená a podpora je zabezpečená aj formou mailing-listu, takže je dostupných mnoho informácií o rád pre programátora.

7.2.3 Genetický algoritmus

Tak ako aj pri predchádzajúcich dvoch moduloch je aj tu vhodné použiť prepracovaný nástroj tretej strany. Vlastná implementácia by síce dodala viac kontroly nad kódom, avšak nebola by zrovnateľná s možnosťami a robustnosťou knižníc tretích strán. Takisto aj kontrolu nad kódom sa pri prehľadne implementovanej knižnici dá po jej naštudovaní na úrovni zdrojového kódu získať. Preto som sa rozhodol využiť pre implementáciu genetického algoritmu externú knižnicu GAlib. Táto knižnica poskytuje mechanizmy genetických algoritmov, ich využitie v podobe kvalitného návrhu je však stále na programátorovi.

Okrem implementácie mechanizmov genetických algoritmov ponúka GAlib aj vysoké množstvo príkladov na úrovni zdrojových kódov, pomocou ktorých sa dá inšpirovať pri vývoji vlastného programu. Pri používaní tejto knižnice sa najčastejšie pracuje s dvoma objektmi:

- *genóm* – objekt reprezentujúci chromozóm, jedinca, ktorý je nositeľom riešenia, pre tento objekt programátor definuje spôsob inicializácie populácie, priebeh mutácie a kríženia,
- *genetický algoritmus* – tento objekt má na starosti celý evolučný proces, od programátora potrebuje zadať ohodnocovaciu funkciu, pravdepodobnosti kríženia, mutácie, a prípadne explicitné definovanie stratégií selekcie a obnovy populácie spolu s číselnými parametrami týchto metód.

Pre implementáciu genetického algoritmu pomocou GAlib je na programátorovi zvoliť čo najlepšiu formu reprezentácie problému. Tu GAlib ponúka štyri dátové typy pre genóm, a to *GAListGenome* kde sú gény uložené vo forme zoznamu, *GATreeGenome* využíva na uloženie génov stromovú štruktúru, typ *GAArrayGenome* bol použitý pri implementácií zadanej aplikácie a ukladá gény do poľa a poslednou možnosťou je reprezentácia chromozómu formou binárneho reťazca pomocou typu *GABinaryStringGenome.* Každý z týchto typov má svoje odvodené podtypy, takže je možné definovať napríklad chromozóm vo forme dvoj-rozmerného poľa, alebo cyklického zoznamu atď. Po zvolení toho správneho objektu pre *genóm* a teda vyriešení otázky reprezentácie problému a definícií inicializácie spolu s operátormi kríženia je potrebné pri vývoji vyriešiť správne nastavenie objektu *genetický algoritmus.*

Pre samotný proces evolúcie je možné zvoliť zo štyroch typov objektu *genetického algoritmu*, ich názvy sú uvádzané použitím terminológie GAlib:

- Genetický algoritmus označovaný názvom *simple* používa na riešenie neprekrývajúce sa populácie, s možným elitizmom. To znamená, že každú generáciu algoritmus vytvorí novú populáciu jedincov, krížením rodičovských jedincov z predchádzajúcej generácie a prípadnou mutáciou.

- *Steady-state* algoritmus používa prekrývajúce sa generácie a je tu treba explicitne určiť aký percentuálny podiel starej generácie bude nahradený novými potomkami.

- *Incremental* je charakteristický tým, že v každej novej generácií sú nahradení potomkami maximálne dvaja rodičia. Toto spôsobuje veľmi pomalú konvergenciu riešenia. Je tu potrebné nastaviť ktorého rodiča budú potomkovia nahrádzať (najslabšieho, najstaršieho, ...).

- Posledným typom je tzv. *deme* genetický algoritmus, ktorý vytvára viacero paralelných populácií.

Pre objekt genetického algoritmu je ako posledné nutné nastaviť ukončovaciu podmienku, ktorá môže byť počet generácií, hodnota chromozómu s najvyššou fitness atp. GAlib poskytuje naviac objekt *GAstatistic*, ktorý zbiera štatistické informácie o celom procese evolúcie. Sú to údaje o najlepšom a najhoršom jedincovi v generácií, počtoch krížení a mutácií a mnoho ďalších.

Kompletná dokumentácia s prehľadne popísaným API pre knižnicu GAlib je dostupná v [38]. Pri popise implementácie zadaného programu sú uvedené výpisy kódu pre genetický algoritmus aj s popisom ich významu.

7.3 Implementácia

Programátorská časť práce za účelom vytvorenia popisovaného programu prebehla ako už bolo spomenuté v jazyku C++. Jej hlavný výstup, preložiteľné zdrojové kódy ako aj výsledná spustiteľná aplikácia so všetkými potrebnými knižnicami sú uvedené ako príloha práce na CD.

Ústrednou témou tejto práce je oblasť genetických algoritmov. Z toho dôvodu pokladám za vhodné, uviesť tu úryvok zo zdrojového kódu, ktorý zachytáva súčasti implementovaného genetického algoritmu.

Prvým krokom bolo definovanie reprezentácie problému, teda zvolenie dátového typu chromozómu, zobrazené vo výpise 7.1. Ďalej tu je zobrazená definícia inicializácie pre populáciu, operácie mutácie a kríženia. Výpis 7.1 teda popisuje všetky komponenty nutné pre objekt *genómu*.

Výpis 7.1: Kód definícii komponent pre objekt genómu.

```
//metóda inicializácie populácie
void initialize(GAGenome &g)
{
  GA1DArrayGenome<double>& genome = (GA1DArrayGenome<double>&)g;
  //incializácia každého jedinca populácie náhodnou hodnotou v zadanom rozmedzí
  for(int i = 0; i < glength; i++){
    genome.gene(i, GARandomDouble(INTERVAL_LOW, INTERVAL_HIGH));
  }
}

//aritmetické kríženie rodičov
int crossover(const GAGenome &parent1, const GAGenome &parent2,GAGenome
*offspring1, GAGenome *offspring2)
{
  double alpha; //náhodný reálny parameter
  for(uint i = 0; i < glength; i++){
    alpha = GARandomDouble(0.0, 1.0);
    offspring1.gene(i,alpha * parent1.gene(i) + ((1.0-alpha)*parent2.gene(i)));
    offspring2.gene(i,(1.0-alpha)*parent1.gene(i) + (alpha * parent2.gene(i)));
    }
    return 1; //návrat 1 kvôli štatistickým účelom
}

//multiplication of gene value in interval 0.9 - 1.1
int mutator(GAGenome &g, double pmut)
{
  int result = 0; //štatistické účely
  double alpha; //náhodná hodnota pre mutáciu génu
  for (int i = 0; i < glength; i++)
  {
    if (GAFlipCoin(pmut)) //genóm bude mutovaný s danou pravdepodobnosťou
    {
      alpha = GARandomDouble(0.9,1.1);
      genome.gene(i, alpha*genome.gene(i));
      result++;
    }
  }
  return result;//počet mutovaných génov v rámci chromozómu
}

//definícia reprezentácie chromozómu, nastavenie počtu génov a konkrétnej
//ohodnocovacej funkcie v závislosti na zvolenej aproximačnej funkcii
GA1DArrayGenome<double> genome(glength, objective[function_type]);
```

Druhou vhodnou ukážkou kódu súvisiacim s genetickým algoritmom je definícia objektu genetického algoritmu a nastavenie parametrov genómu. Tento úsek kódu zachytáva výpis 7.2.

Výpis 7.2: Kód nastavenia a spustenia genetického al

```
GA1DArrayGenome<double> genome(glength, objective[function_type]);
//inicializácia generátora náhodných čísel
GARandomSeed();
//priradenie funkcie na inicializáciu jedinca
genome.initializer(::initialize);
//nastavenie funkcie mutácie
genome.mutator(::mutator); // set the function for mutation

//vytvorenie objektu genetického algoritmu a nastavovanie jeho parametrov
GASteadyStateGA ga(genome);
//spôsob kríženia rodičovských jedincov
ga.crossover(::crossover);
//nastavenie veľkosti populácie - počtu jedincov v populácií
ga.populationSize(popsize);
//ukončenie algoritmu po dosiahnutí ngens generácií
ga.nGenerations(ngens);
//pravdepodobnosť kríženia
ga.pCrossover(pcross);
//pravdepodobnosť mutácie
ga.pMutation(pmut);
//percentuálny podiel rodičovskej populácie nahradenej potomkami
ga.pReplacement(0.6);
//spustenie evolučného procesu
ga.evolve();
```

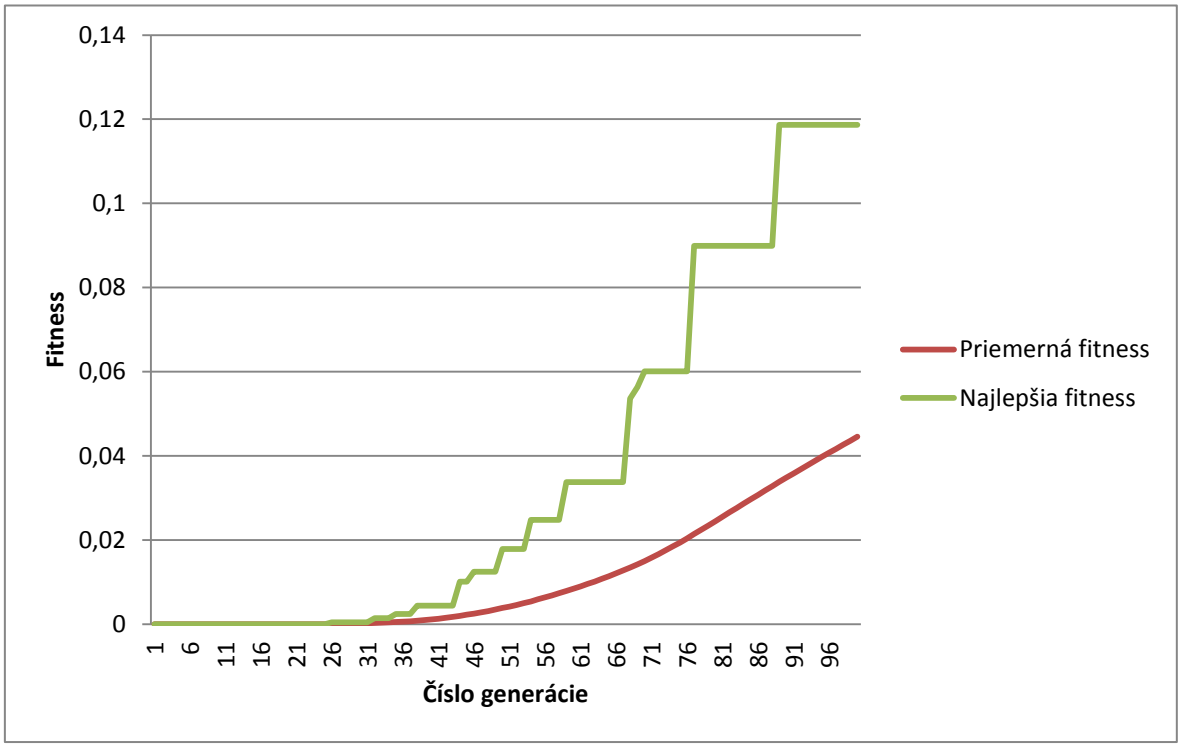
Testovanie korektnosti výsledkov finálnej aplikácie bolo zjednodušené grafickým zobrazením výsledkov, kedy je viditeľné, že výsledné krivky sú naozaj spojité a aproximujú dané body. Pretože genetické algoritmy z princípu neposkytnú optimálne riešenie, je vyhodnotenie optimálnosti výsledkov programu skôr na užívateľovom posúdení prípadu od prípadu.

Funkčné testovanie som previedol klasickým spôsobom testovania každej logickej vetvy programu, nasimulovaním čo najviac chybových stavov, ako sú nekorektné hodnoty konfiguračného súboru, zadané body mimo definičného oboru aproximovanej funkcie a podobne. Z dôvodu prílišnej rozsiahlosti tu neuvádzam jednotlivé prevádzané testovacie scenáre.

7.4 Parametre implementovaného genetického algoritmu

Po implementácií aplikácie bolo možné experimentovať s nastavovaním parametrov genetického algoritmu. Jednalo sa hlavne o manipuláciu s počtom jedincov v populácii a počtom generácii, pri ktorých dosiahnutí algoritmus ukončí svoju činnosť a jedinec z poslednej populácie s najvyššou fitness bude zodpovedať nájdenému riešeniu. Nastavenie všetkých parametrov genetického algoritmu je umožnené previesť aj užívateľovi. Počiatočné a odporúčané hodnoty nastavovaných parametrov však boli odvodené empirickým experimentovaním s výsledným programom. Nasledujúce grafy zobrazujú konvergenciu riešenia, teda fitness hodnotu najlepšieho jedinca populácie, a priemernú hodnotu fitness v závislosti od počtu generácií. Za ním nasledujú variácie tejto závislosti, so zvyšovaním veľkosti populácie. V zobrazovanom testovacom prípade bolo zadávaných sedem nameraných bodov pre každý interval určených ku aproximácií. Zobrazované sú výsledky aproximácie polynomiálnou funkciou štvrtého rádu, pretože táto má najvyššie časové nároky na výpočet.

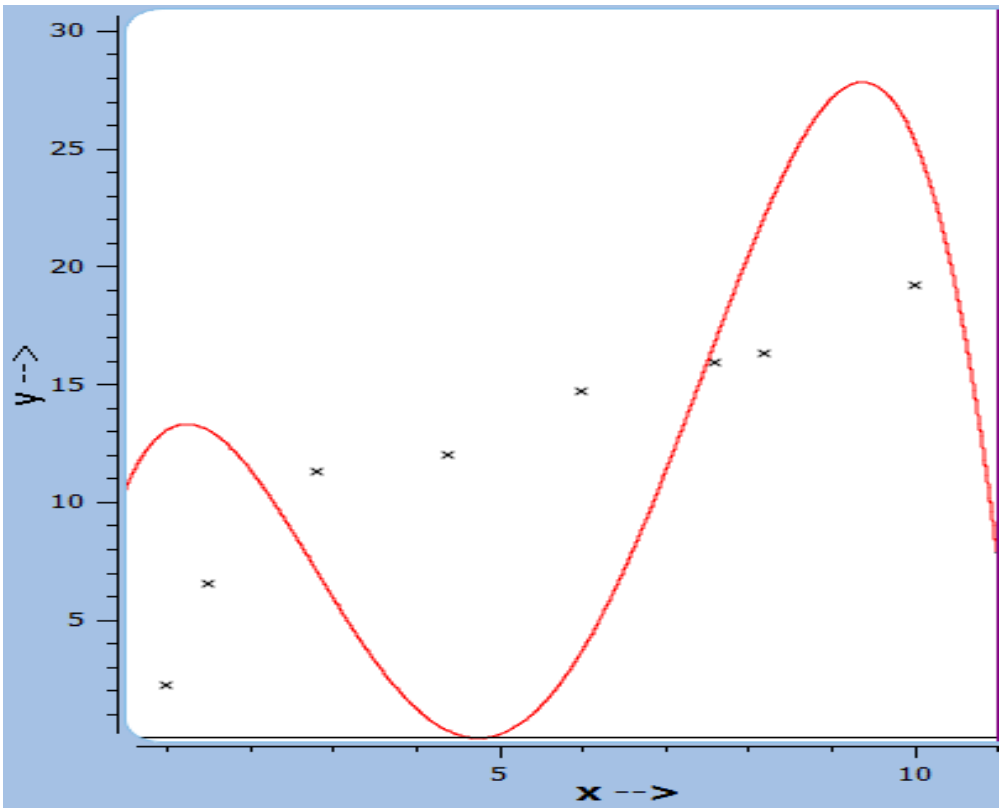
Obrázok 7.1 zobrazuje vývoj fitness hodnoty populácie o veľkosti 500 jedincov, počas vývoja 100 generácií a hľadajúcich riešenie pre aproximáciu polynomiálnej aproximácie štvrtého stupňa.



Obrázok 7.1:
Vývoj fitness pre veľkosť populácie 500 a počet generácií 100

Na obrázku je vidno trend naznačený v predchádzajúcich kapitolách, kde priemerná fitness pomaly dosahuje hodnoty najlepšieho jedinca, tento proces je tu v dôsledku vysokého poč-

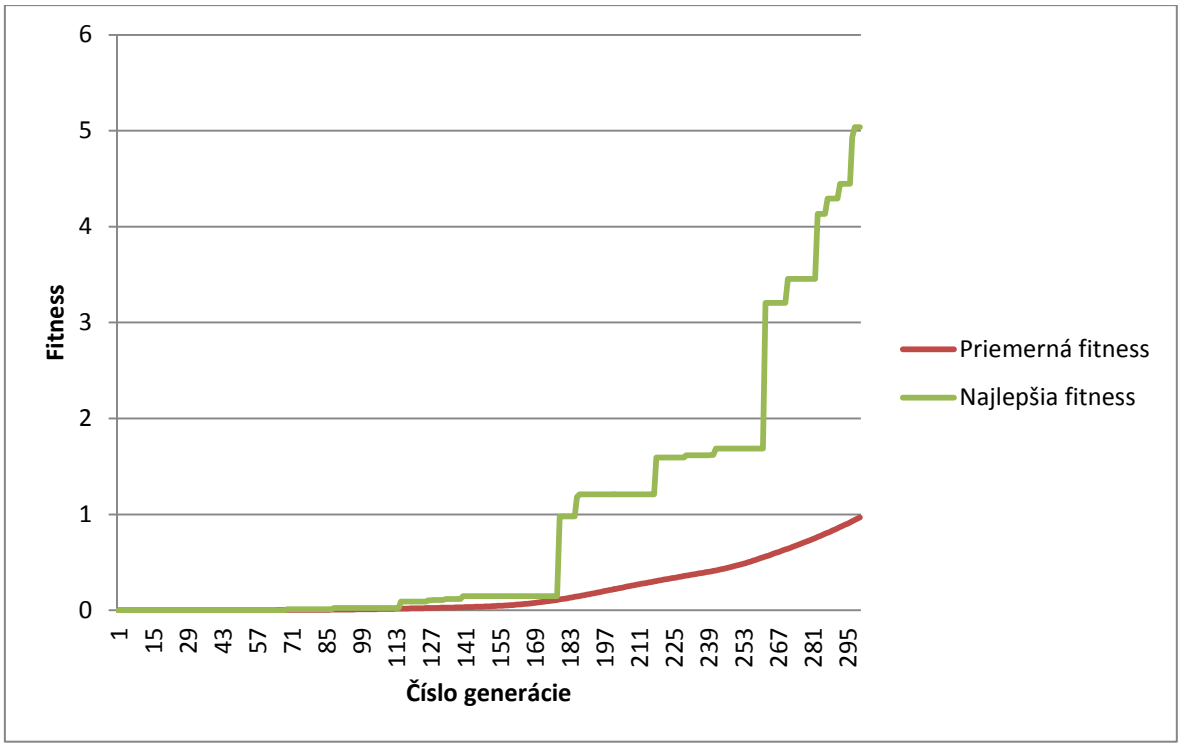
tu jedincov viac skokový ako plynulý. Obrázok 7.2 ukazuje vizuálny výstup riešenia, ktorého vývoj fitness zachytáva obrázok 7.1.



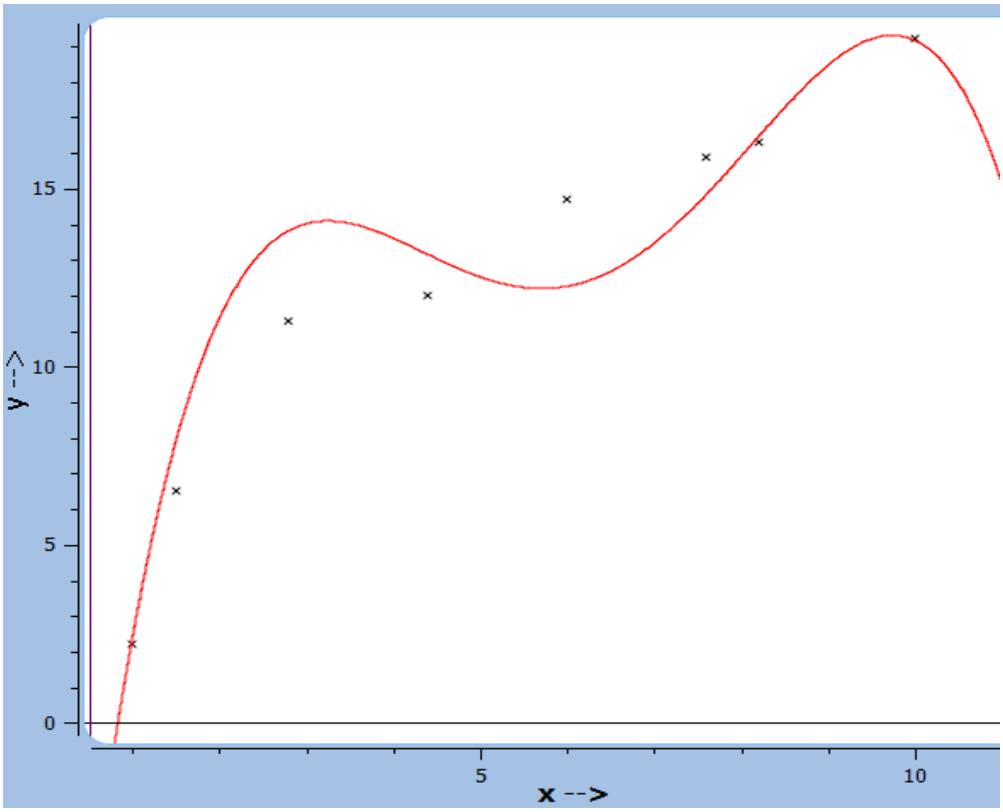
Obrázok 7.2:
Zobrazenie polynomiálnej aproximácie štvrtého stupňa pre 500 jedincov a 100 generácií

Po prevedení niekoľkých cyklov opakovania výpočtu s rôznymi hodnotami veľkosti populácie a počtu generácií som dospel ku najvhodnejšej počiatočnej hodnote týchto dvoch parametrov, a tou je tisíc jedincov v populácií a tristo iterácií genetického algoritmu. Algoritmus s týmito hodnotami je stále relatívne rýchly, kedy exekúcia trvá v ráde pár sekúnd, a výsledky sú viac než uspokojujúce. Konvergencia fitness a vizuálna podoba riešenia je na obrázku 7.3, resp. 7.4.

Je samozrejmé, že so zvyšovaním počtu jedincov a generácií bude stúpať aj presnosť riešenia, a aj preto je umožnené explicitné zadanie týchto parametrov pomocou konfiguračného súboru užívateľovi. Zvyšné pôvodné parametre genetického algoritmu zodpovedajú hodnotám najčastejšie odporúčaným v literatúre, pravdepodobnosť mutácie je jedno percento, pravdepodobnosť kríženia je deväťdesiat percent a do novej generácie prechádza šesťdesiat percent potomkov.



Obrázok 7.3:
Vývoj fitness pre veľkosť populácie 1000 a počet generácií 300

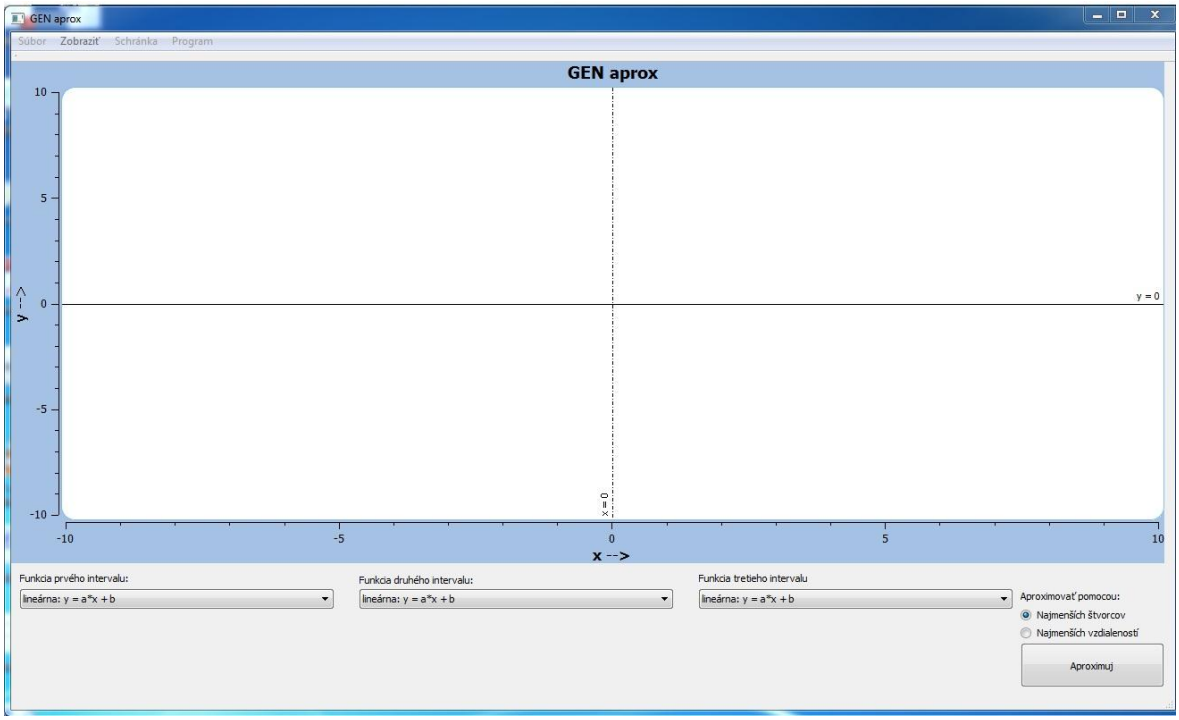


Obrázok 7.4:
Zobrazenie polynomiálnej aproximácie štvrtého stupňa pre 1000 jedincov a 300 generácií

7.5 Ukážka programu

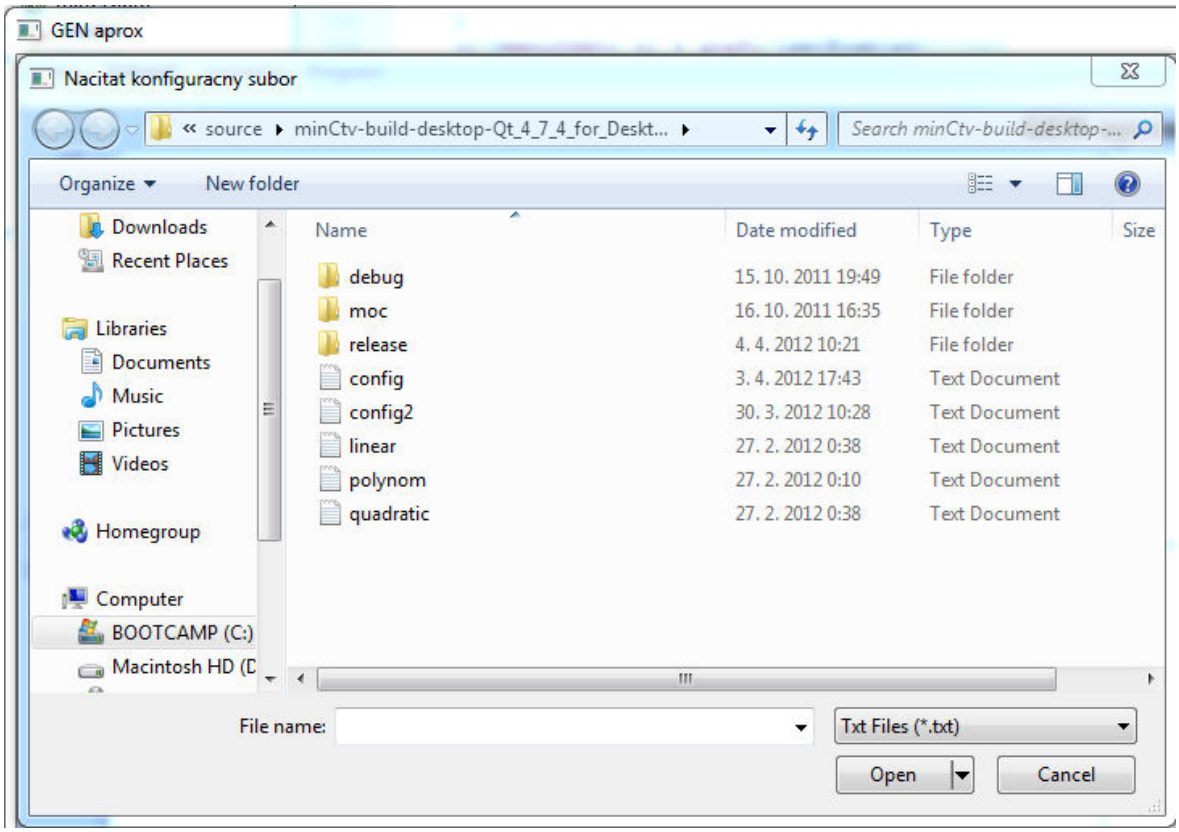
Táto časť obsahuje extrakt z užívateľskej príručky, ktorej kompletná podoba je uvedená v prílohe práce. Názov programu je Gen aprox.

Grafická podoba výslednej aplikácie je zachytená na obrázku 7.5.

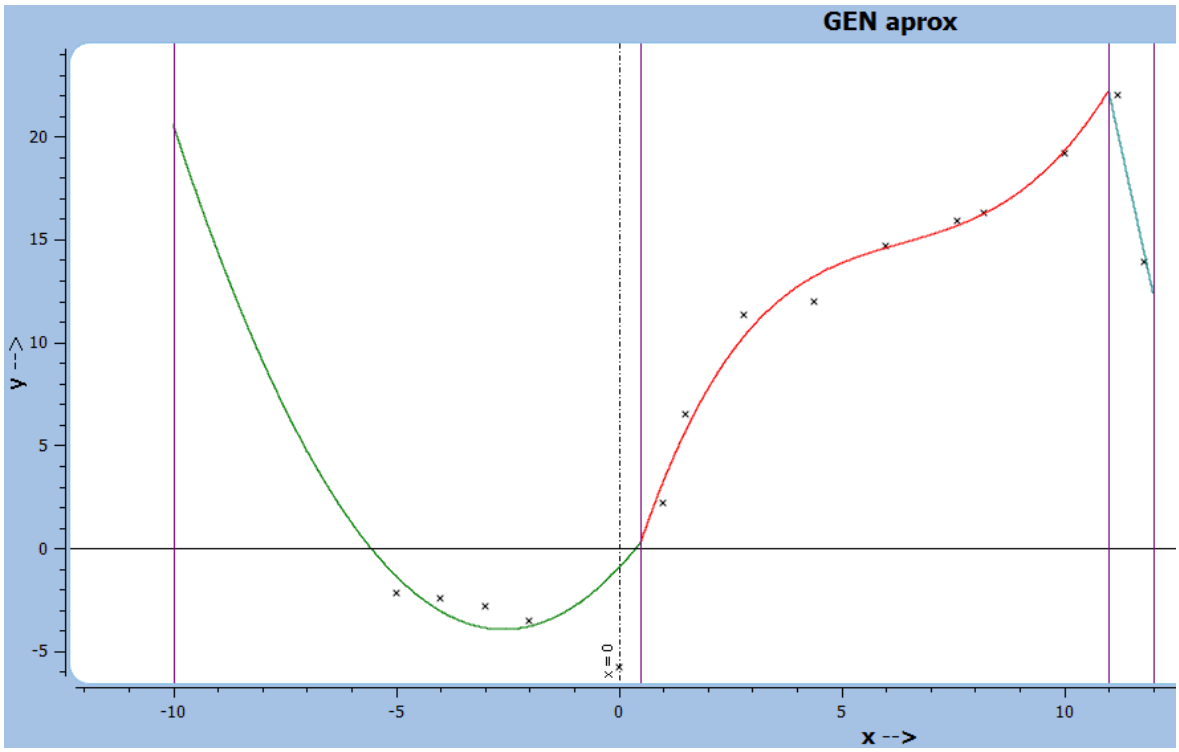


Obrázok 7.5:
Výsledná podoba aplikácie

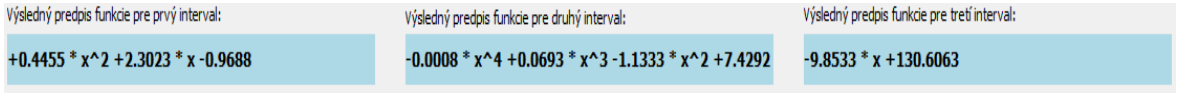
Načítanie vstupného konfiguračného súboru prebieha pomocou dialógu, ktorý je zobrazený na obrázku 7.6. Výstup programu formou grafu ako aj analytického predpisu výsledných aproximačných funkcií je na obrázkoch 7.7 a 7.8.



Obrázok 7.6:
Dialóg na otvorenie konfiguračného súboru



Obrázok 7.7:
Grafická podoba výstupu



Obrázok 7.8:
Analytické predpisy výsledných funkcií

Ovládanie programu je veľmi intuitívne. Podrobný popis programu a kompletný návod na použitie je uvedený v prílohe A, takisto je text v prílohe distribuovaný s výslednou aplikáciou.

8 Záver

Hlavným cieľom tejto práce bolo dodať program slúžiaci na aproximáciu empiricky nameraných bodov nelineárnymi matematickými funkciami pomocou metódy najmenších štvorcov s využitím genetických algoritmov. Požiadavky na aplikáciu boli sformulované Fyzikálnym ústavom Akadémie vied ČR, s tým, že výsledná aplikácia by mohla byť využívaná viacerými zahraničnými vedeckými ústavmi.

Na vytvorenie požadovanej aplikácie som previedol niekoľko krokov, ktoré sú bližšie popísané v práci. Prvým krokom bol prieskum prác s podobným zameraním. Tu bolo zistené, že neexistuje dostupná aplikácia vyhovujúca zadaniu. Preto bolo nutné vytvoriť celkom novú aplikáciu spĺňajúca všetky požiadavky definované zadaním.

Druhým krokom bolo teoretické naštudovanie problematiky genetických algoritmov a metódy najmenších štvorcov. Pre aproximáciu bola ďalej navrhnutá metóda najmenších vzdialeností, ktorá nepočíta s druhou mocninou vzdialenosti bodu od priamky, preto je aj tolerantnejšia voči náhodným chybám merania. Pri návrhu genetického algoritmu som dospel k záveru, že najlepšou kombináciou genetických algoritmov a aproximačných metód je použitie metódy najmenších štvorcov, resp. najmenších vzdialeností ako ohodnocovacích funkcií, určujúcich fitness hodnotu každého jedinca. Pre svoju názornosť môže slúžiť popis návrhu genetického algoritmu ako šablóna pre návrh genetických algoritmov riešiacich širokú paletu úloh.

Pri samotnej implementácií bol z dôvodu dostupnosti najvhodnejších vývojových prostriedkov, nástrojov pre vykresľovanie grafov a knižníc pre prácu s genetickými algoritmami použitý jazyk C++. Pre širšiu použiteľnosť práce sú tu uvedené úryvky zdrojových kódov týkajúce sa implementácie genetického algoritmu, a tak môže táto práca slúžiť ako návod pre programátorov pri vývoji zložitejších systémov používajúcich genetické algoritmy.

Vyvinutý program bude potenciálne používaný Fyzikálnym ústavom AV ČR a prípadné požiadavky na zmenu budú riešené operatívne. Program je distribuovaný ako voľne stiahnuteľná a ďalej šíriteľná aplikácia prostredníctvom internetu. Spustiteľná aplikácia, ako aj zdrojové kódy aplikácie v jazyku C++ a užívateľská príručka tvoria prílohu práce.

Teoretickú časť práce by bolo možné rozšíriť o formálne dôkazy korektnosti nájdeného riešenia genetickým algoritmom, tzv. teorém schém, prípadne o širší rozbor matematických aproximačných metód ako aj o formálne odvodenie metódy najmenších štvorcov pomocou parciálnych derivácií.

Program, implementovaný ako súčasť práce je možné rozšíriť o interaktívne zadávanie vstupných bodov užívateľom a vylepšiť dizajnérsku stránku grafického rozhrania. Program bol však implementovaný tak, aby bol jednoducho rozšíriteľný a jeho vývoj nekončí, ale bude postupne, na základe pripomienok užívateľov vylepšovaný.

Terminologický slovník

Termín	Význam [zdroj]
Optimalizácia	Proces výberu najlepšieho variantu z množstva možných javov [39]
Najoptimálnejšie	V kontexte práci sa myslí riešenie, najviac sa približujúcemu tomu optimálnemu, pokiaľ sa hovorí o viacerých riešeniach približujúcich sa k optimu, môže byť jedno optimálnejšie ako druhé [autorova definícia]
Hill climbing	Matematická optimalizačná technika, vyhľadáva lokálne extrémy [28]
Iterácia	Opakovanie [39]
Definičný obor funkcie	Interval, na ktorom je funkcia definovaná [autorova definícia]
Binárna reprezentácia	Dvojčlenné zastúpenie, v kontexte práce sa myslí reprezentácia formou dvojkovej sústavy, teda symbolov 1 a 0 [autorova definícia]
Aproximácia	Priblíženie, nahradenie čísla najbližším vhodným číslom [39]
Freeware	Voľne šírený počítačový program bez platenia autorského honoráru [39]
Exe súbor	Koncovka spustiteľného súboru na platforme Windows [autorova definícia]
Skriptovací jazyk	Interpretovaný programovací jazyk s vyššou mierou abstrakcie príkazov [autorova definícia]
Skript	Program napísaný v skriptovacom jazyku [autorova definícia]
Ruby	Konkrétny zástupca skriptovacieho jazyka [autorova definícia]
Rails	Podporná softvérová štruktúra slúžiaca na uľahčenie vývoja webových aplikácií [autorova definícia]
Drag & drop	Systém práce s komponentmi rozhrania zvolením konkrétnej komponenty z kontajnera a jej pretiahnutie a umiestnenie na plochu výsledného rozhrania [autorova definícia]
Toolkit	Predpripravená sada nástrojov [40]
Makefile	Zdrojový kód pre kompilátor s údajmi o vzťahoch medzi kompilovanými súbormi, použitými knižnicami a parametrami prekladu [autorova definícia]
Štandardná knižnica jazyka C++	Súbor príkaz jazyka C++, ktoré sú definované štandardom ISO [autorova definícia]
Parsovanie	Proces prechádzania textového súboru a vyhľadávania kľúčových slov za vopred definovaných podmienok [autorova definícia]

Zoznam literatúry

[1] Grefenstette, J.J. (Editor), *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985. [cit. 01.03.2011]

[2] DeJong, K.A. *Genetic Algorithms: A 10 Year Perspective.* V [1], pp. 169-177. [cit. 01.03.2011]

[3] Schwefel, H.-P. *Numerical Optimization for Computer Models,* John Wiley, Chichester, UK, 1981. [cit. 01.03.2011]

[4] Hauser, J. W, a C. N Purdy. *Designing a Genetic Algorithm for Function Approximation for Embedded and ASIC Applications*. V *49th IEEE International Midwest Symposium on Circuits and Systems,* 2006. MWSCAS '06, 2:555–559. IEEE, 2006. . [cit. 01.03.2011]

[5] Hauser, James, a Carla Purdy. *Approximating Nonlinear Functions with Genetic Algorithms*. *EE Times*, [online] [cit. 03.03.2011] Dostupné z: http://www.eetimes.com/design/embedded/4024501/Approximating-Nonlinear-Functions-with-Genetic-Algorithms.

[6] Hauser, Jim. *Function Approximation Using a Genetic Algorithm*, [online] [cit. 03.03.2011] Dostupné z: http://secs.ceas.uc.edu/~cpurdy/jimhauser/jhauser.htm.

[7] Svoboda, Petr. *Hledání parametrů nelineárních funkcií genetickým algoritmem,* ČVUT v Praze, fakulta elektrotechnická, Katedra řídící techniky, 2009. , [online] [cit. 03.03.2011] Dostupné z: http://www.poppik.net/prace/geneticke_algoritmy.pdf

[8] RNDr. Vladimír Tichý. "MNC - Vladimír Tichý", [online] [cit. 03.03.2011] Dostupné z: http://nb.vse.cz/~tichy/MNC.htm

[9] L. Davis, S. Coombs. *Genetic algorithms and communication link speed design: theoretical considerations.* V Grefenstette, J.J., (Editor), *Proceedings of the Second International Conference on Genetic Algorithms,* Lawrence Erlbaum Associates, Hillsdale, NJ, 1985, pp. 252 – 256. [cit. 03.03.2011]

[10] Potter, W.D., Miller, J.A, Tonn, B.E., Gandham, R.V. and Lapena C.N. *Improving the reliability of heuristic multiple fault diagnosis via ec-based genetic algorithm.* Applied Intelligence, 2:5-24, 1992. [cit. 03.03.2011]

[11] Peter Ross, Dave Corne, Hsiao-Lan Fang. *Improved evolutionary timetabling through delta evaluation and directed mutation.* V Davidor, Y., Schwefel, H-P., *Parallel Problem Solving from Nature III*. Springer-Verlag, 1994. [cit. 03.03.2011]

[12] Rahmani, A.T., Ono, N. *A genetic algorithm for channel routing problem.* V Forest, S. (Editor), *Proceedings of ICGA-93,* Morgan Kaufmann, San Mateo, Ca., 1993*, pp. 494-498*. [cit. 03.03.2011]

[13] Davidor Y., *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization.* World Scientific, Singapore, London, 1990. [cit. 04.03.2011]

[14] Schulze-Kremmer Steffen. *Protein Folding Simulation by Force Field Optimisation.* In: *Genetic algorithms and protein folding,* [online] [cit. 04.03.2011] Dostupné z: http://www.techfak.uni-bielefeld.de/bcd/Curric/ProtEn/121.html

[15] Grefenstette, J.J. *Strategy acquisition with genetic algorithms.* V Davis, L., (Editor), *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, 1991, pp. 186-201. [cit. 04.03.2011]

[16] Bauer, Richard J. *Genetic Algorithms and Investment Strategies.* John Wiley and Sons, 1994, ISBN 9780471576792 [cit. 04.03.2011]

[17] Nesnídal, Tomáš. "Genetické algoritmy: K čemu jsou pro trading dobré? - Financnik.cz". *Finančník*, [online] [cit. 04.03.2011] Dostupné z: http://www.financnik.cz/komodity/fin_home/geneticke-algoritmy-a-trading-k-cemu-jsou-dobre.html

[18] Bidlo M.: Evolutionary Design of Generic Structures Using Instruction-Based Development, Brno University of Technology, 2010, p. 124 [cit. 04.03.2011]

[19] "15 Real-World Applications of Genetic Algorithms". *brainz*, [online] [cit. 04.03.2011] Dostupné z: http://brainz.org/15-real-world-applications-genetic-algorithms/

[20] Darwin, C. *On the Origin of Species By Means of Natural Selection*. Dover publications, 2006. [cit. 04.03.2011]

[21] Eiben, Agoston E., a James E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003 ISBN 9783540401841 [cit. 05.03.2011]

[22] Aliev,R.A, Aliev,R.R.: Soft Computing and its Application, World Scientific Publishing Co. Pte. Ltd., 2001, ISBN 981-02-4700-1, [online] [cit. 05.03.2011] Dostupné z: http://books.google.cz/books?id=C05X9svt8WQC&printsec=frontcover&hl=cs&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[23] Sivanandam, S.N., a S. N. Deepa. *Introduction to Genetic Algorithms*. 1st edition. Springer, 2007. ISBN 354073189X [cit. 05.03.2011]

[24] V. Kvasnička, J. Pospíchal a P. Tiňo. *Evolučné algoritmy*. Vydavateľstvo STU, Bratislava, 2000. [cit. 06.03.2011]

[25] Bäck, T., Fogel, D.B., Michalewicz Z., Eds. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publishing, Bristol, 2000 [cit. 06.03.2011]

[26] Beasley, David, David R Bull, a Ralph R Martin. *An Overview of Genetic Algorithms: Part 1, Fundamentals*, 1993.[cit. 07.03.2011]

[27] Hynek, Josef. *Genetické algoritmy a genetické programování*. Grada Publishing a.s., 2008. ISBN 9788024726953 [cit. 07.03.2011]

[28] Mitchell, Melanie. *An Introduction to Genetic Algorithms*. Third Printing. A Bradford Book, 1998. ISBN 0262631857 [online] [cit. 07.03.2011]Dostupné z: http://www.boente.eti.br/fuzzy/ebook-fuzzy-mitchell.pdf

[29] Eiben, Agoston E., a James E. Smith. "Introduction to Evolutionary Computing", [online] [cit. 07.03.2011] Dostupné z: http://www.cs.vu.nl/~gusz/ecbook/ecbook-illustrations.html

[30] Bretscher, Otto (1995). Linear Algebra With Applications, 3rd ed.. Upper Saddle River NJ: Prentice Hall. [cit. 10.03.2011]

[31] *Doing Least Squares: Perspectives from Gauss and Yule*. International Statistical Review 66, [online] [cit. 10.03.2011] Dostupné z: http://www.economics.soton.ac.uk/staff/aldrich/yule%20gauss.pdf

[32] MathBits. "Statistics 2 - Least Squares", [online] [cit. 10.03.2011] Dostupné z: http://mathbits.com/mathbits/tisection/statistics2/leastsquares.htm

[33] Wolfram Research, Inc. "least squares - Wolfram Search", [online] [cit. 10.03.2011] Dostupné z: http://mathworld.wolfram.com/search/?query=least+squares&x=0&y=0

[34] Nokia Corporation. "Online Reference Documentation", [online] [cit. 25.03.2011] Dostupné z: http://doc.qt.nokia.com/

[35] Summerfield, M. Advanced Qt Programming: *Creating Great Software with C++ and Qt 4 (1st ed.)*, Addison-Wesley, p. 550, ISBN 978-0321635907. [cit. 25.03.2011]

[36] "Jellycan Code - SimpleIni", [online], 2012 [cit. 16.03.2011] Dostupné z: http://code.jellycan.com/simpleini/

[37] "Qwt User's Guide: Qwt - Qt Widgets for Technical Applications" [online], 2012 [cit. 25.03.2011]Dostupné z: http://qwt.sourceforge.net/index.html

[38] Law, Matthew. "GAlib: Matthew's C++ Genetic Algorithms Library", [online], 2012 [cit. 25.03.2011] Dostupné z: http://lancet.mit.edu/galib-2.4/

[39] "Slovník cudzích slov", [online], 2012 [cit. 29.03.2011] Dostupné z: http://www.slovnikslov.sk/

[40] "PC slovník", [online], 2012 [cit. 29.03.2011] Dostupné z: http://www.klasici.sk/old/publikovanie/slovnik/pc_slovnik.html.

Zoznam obrázkov

Obrázok 2.1: Uhly α, β, γ, ktoré genetický algoritmus vypočíta pre optimálny presun robotického ramena z bodu X do bodu Y 13
Obrázok 3.1: Vzťah medzi evolučným procesom v prírode a riešením úloh genetickými algoritmami 16
Obrázok 3.2: Zaradenie genetických algoritmov do oblasti soft computingu 17
Obrázok 3.3: Zaradenie genetických algoritmov do kontextu optimalizačných metód 18
Obrázok 3.4: Zobrazenie vzťahu medzi génom, chromozómom a populáciou v oblasti genetických algoritmov 20
Obrázok 3.5: Zobrazenie genotypu farby pomocou modelu CMY na fenotyp 20
Obrázok 3.6: Základná schéma genetického algoritmu 21
Obrázok 3.7: Typický priebeh fitness funkcie 24
Obrázok 3.8: Rozdelenie ruletového kolesa pre hodnoty fitness 10, 4, 4, 2 25
Obrázok 3.9: Príklad n-bodového kríženia pre n=2 27
Obrázok 3.10: Príklad uniformného kríženia 27
Obrázok 3.11: Príklady rôznych metód mutácie 29
Obrázok 4.1: Ilustrácia princípu použitých matematických metód 33
Obrázok 5.1: Use case diagram systému 38
Obrázok 6.1: Základný náčrt grafického užívateľského rozhrania 44
Obrázok 6.2: Základný náčrt menu aplikácie 45
Obrázok 6.3: Základný náčrt panelu pre užívateľské ovládanie programu 45
Obrázok 6.4: Príklad populácie o veľkosti 4 pre matematickú funkciu s tromi parametrami 52
Obrázok 6.5: Zobrazenie fenotypu pre krivku reprezentovanú jedincom s nízkou fitness. 54
Obrázok 6.6: Zobrazenie fenotypu pre krivku reprezentovanú jedincom s vysokou fitness 55
Obrázok 7.1: Vývoj fitness pre veľkosť populácie 500 a počet generácií 100 68
Obrázok 7.2: Zobrazenie polynomiálnej aproximácie štvrtého stupňa pre 500 jedincov a 100 generácií 69
Obrázok 7.3: Vývoj fitness pre veľkosť populácie 1000 a počet generácií 300 70
Obrázok 7.4: Zobrazenie polynomiálnej aproximácie štvrtého stupňa pre 1000 jedincov a 300 generácií 70
Obrázok 7.5: Výsledná podoba aplikácie 71
Obrázok 7.6: Dialóg na otvorenie konfiguračného súboru 72
Obrázok 7.7: Grafická podoba výstupu 72
Obrázok 7.8: Analytické predpisy výsledných funkcií 73
Obrázok A.1: Dialóg na otvorenie konfiguračného súboru 87
Obrázok A.2: Ovládanie spustenia výpočtu 88
Obrázok A.3: Grafický výstup programu - graf 91
Obrázok A.4: Grafický výstup programu – analytický predpis 91

Príloha A: Užívateľská príručka

A.1 Úvod

Pre využitie tohto popisu je predpokladaná základná znalosť práce s PC a orientácie v základnej terminológií. Vhodná je znalosť princípov metódy najmenších štvorcov ako aj výpočtov pomocou genetického algoritmu, nie je to však nutný predpoklad pre plnohodnotné využívanie programu Gen aprox.

Program Gen aprox bol vytvorený na základe zadania Fyzikálneho ústavu Akadémie vied ČR. Jeho funkcionalita spočíva v aproximovaní vopred zadaných empiricky nameraných bodov na požadovaná matematickú funkciu. Výstup programu je rozdelený na tri intervaly, kde je pre každý interval potrebné zvoliť aproximačnú funkciu a následný výsledný graf je na vymedzenom priestore tromi intervalmi spojitý.

Autorom programu je Bc. Matúš Holec, študent posledného ročníka Katedry informačních technologií, Fakulty informatiky a statistiky VŠE v Prahe. Program vznikol ako súčasť diplomovej práce a je voľne šíriteľný.

A.2 Definícia pojmov

Najprv je nutné definovať pojmy a skratky, ktoré budú v texte používané.

GA – genetický algoritmus, tvorí jadro výpočtu a je hlavnou komponentou zodpovednou za dodanie správneho výsledku, užívateľskú interakciu tvorí nastavovanie niektorých parametrov GA.

MNŠ – metóda najmenších štvorcov, tvorí vyhodnocovanie generovaných aproximačných funkcií, česká zaužívaná skratka je MNČ a anglická LSM (least square method)

MNV – druhá metóda vyhodnocovania vhodnosti funkcií, metóda najmenších vzdialeností, nepočíta sa tu s druhou mocninou y-ovej vzdialenosti bodu od priamky ale s absolútnou hodnotou tejto vzdialenosti.

Panel užívateľskej interakcie – miesto na grafickom užívateľskom rozhraní, kde môže užívateľ manipulovať a ovplyvňovať beh programu.

Konfiguračný súbor – textový súbor, obsahujúci parametre na nastavenie behu programu.

INI formát – špeciálny formát vstupného konfiguračného súboru v tvare parameter = hodnota, parametre môžu byť uvádzané v rámci sekcií, ktorých rozsah je uvedený v podobe

[sekcia1] až po uvedenie ďalšieho symbolu v hranatých zátvorkách, uvádzajúceho novú sekciu

GUI – grafické užívateľské rozhranie.

Interval výpočtu – výstupný graf tvoria tri funkcie v troch intervaloch zadaných užívateľom.

Roletové menu – menu, po ktorého rozkliknutí užívateľom sa zjavia možnosti výberu funkcií.

A.3 O programe

A.3.1 Inštalácia

Program Gen aprox sa nijak neinštaluje. Stačí do jedného adresára rozbaliť archív s programom a dvojklikom na súbor GENaprox.exe bude program spustený. Prítomnosť všetkých knižníc v komprimovanom archíve je pre beh programu nutná. Prácu s programom uľahčí, ak bude textový konfiguračný súbor umiestnený v rovnakom adresári ako program. Odinštalovanie programu je možné jeho zmazaním.

A.3.2 Obmedzenia

Po spustení výpočtu je nutné rátať s tým, že v závislosti od počtu zadaných bodov, zvolenej aproximačnej funkcie a nastavených parametrov GA môže výpočet trvať aj niekoľko minút.

Ďalej je nutné rátať s druhým obmedzujúcim faktorom a tým je spojitosť aproximovaných funkcií na prelome intervalov, ktorá môže spôsobiť menej presnú aproximáciu zadaných bodov. Teda spojitosť funkcií je na úkor presnosti aproximácie.

A.3.3 Súbory

Nasleduje zoznam súborov, ktoré sú dodávané spolu s programom:

- GENaprox.exe – spustiteľný súbor, hlavný program,
- ga.dll – knižnica zabezpečujúca prácu s genetickým algoritmom,
- libgcc_s_dw2-1.dll – knižnica spojená s prekladačom,

- mingwm10.dll – knižnica spojená s prekladačom,
- QtCore4.dll – podporné funkcie vývojového prostredia,
- QtGui4.dll – knižnica s prvkami GUI,
- QtSvg4.dll – knižnica používaná na spustenie aplikácie,
- qwt.dll – knižnica pre vykreslenie grafu,
- config.txt – vzorový konfiguračný súbor,
- navod_GENaprox.pdf – táto užívateľská príručka.

Zároveň sú pre pokročilejšieho užívateľa dostupné zdrojové kódy aplikácie v jazyku C++, takže je možné upraviť chovanie aplikácie zásahom do zdrojového kódu. Sú to tieto súbory:

- Hlavičkové súbory – súbory obsahujúce deklarácie tried a ich metód:
    - enums.h – použité výčty,
    - genetic_approximator.h – hlavičkový súbor pre modul implementujúci genetický algoritmus,
    - mainwindow.h – deklarácie metód pre modul implementujúci GUI,
    - minctv_configurator.h – hlavičkový súbor pre modul starajúci sa o načítanie vstupného konfiguračného súboru,
    - plotter.h – hlavičkový súbor pre modul vykresľujúci graf,
    - external/SimpleIni.h – externá knižnica použitá na parsovanie INI súborov.
- Zdrojové súbory – súbory obsahujúce definície metód:
    - genetic_approximator.cpp – modul implementujúci genetický algoritmus,
    - main.cpp – modul obsahujúci funkciu main,
    - mainwindow.cpp – obsluha GUI,
    - minctv_configurator.cpp – načítanie konfiguračného súboru v INI formáte,
    - plotter.cpp – metódy vykresľujúce grafy.
- minCtv.pro – konfiguračný súbor pre vývojové prostredie Qt.

Pri manipulácií s implementáciou je v súbore minCtv.pro nutné zmeniť cestu ku súboru qwt.dll a ga.dll podľa toho, kde sú uložené (tieto súbory sú súčasťou archívu so spustiteľnou aplikáciou).

A.4 Ovládanie programu

A.4.1 Vstupné dáta – konfiguračný súbor

Hlavným vstupom programu je konfiguračný súbor s parametrami pre výpočet. Popis tohto súboru je uvedený v tejto časti. Interaktívne zadávanie bodov by bolo príliš nepresné, preto je z funkcionality vypustené. Okrem súradníc bodov určených na aproximáciu je programu možné nastaviť parametre, ovplyvňujúce výpočet.

Formát konfiguračného súboru je formát INI. Špecifikom takéhoto formátu je zadávanie vstupných parametrov v tvare parameter = hodnota_parametra. Špecifikom je možnosť zadávania desatinnej hodnoty v tvare čísla s desatinnou bodkou aj čiarkou. Táto skutočnosť bude demonštrovaná aj v príkladoch.

Popis štruktúry konfiguračného súboru formátu INI:

```
//komentár sa píše za znaky „//“, takto bude popísaná základná štruktúra INI

//súbor môže byť rozdelený na viacero sekcii

//označenie prvej sekcie
[sekcia_jeden]

//hodnota sa premennej priraďuje spôsobom kľúč = hodnota
premenna_x = 23.3
premennaY = true
premenna_z = „obsah“

//ďalšia sekcia
[sekcia_dva]

//v tejto sekcii je možné použiť názvy premenných z iných sekcií
premenna_x = 12
//prípustná je aj viacnásobná definícia premenných
premenna_x = 10
premenna_x = 17
```

Nasledujúci popis formátu konfiguračného súboru sa dodržiava terminológie INI formátu.

Sekcie konfiguračného súboru

Program očakáva na vstupe v konfiguračnom súbore päť sekcií, označené nasledujúcim spôsobom:

- `[section_one]` – táto sekcia obsahuje body a typ aproximovanej funkcie prvého intervalu,

- `[section_two]` – body a typ aproximovanej funkcie druhého intervalu,
- `[section_three]` – parameter pre tretí interval,
- `[general]` – nešpecifické parametre vzhľadom na interval, nastavenie hraníc intervalov, rozsah pôvodného zobrazenia na osách,
- `[genetic]` – parametre GA, jediná nepovinná sekcia.

Niektoré parametre konfiguračného súboru sú povinné, iné sú nepovinné.

Parametre konfiguračného súboru

Tieto parametre program očakáva v sekcii **general**:

- `int1` – povinný parameter určujúci ohraničenie prvého intervalu zľava, jeho hodnota zodpovedá x-ovej hodnote najľavejšej hranice obmedzujúcej výpočet, pr.: `int1 = -1.5,`
- `int2` – povinný parameter vymedzujúci koniec prvého intervalu a zároveň začiatok druhého intervalu, pr.: `int2 = 2,8`,
- `int3` – povinný parameter vymedzujúci koniec druhého intervalu a zároveň začiatok tretieho intervalu, pr.: `int3 = 19,345`,
- `int4` – povinný parameter určujúci koniec tretieho intervalu, zároveň vymedzuje pravú x-ovú hranicu miesta výpočtu, pr.: `int4 = 29.15`,

Parametre určujúce rozsah súradnicových osí pre finálne zobrazenie výsledku (plocha grafu je zoomovateľná):

- `x_from` – nepovinný parameter, jeho hodnota zodpovedá, od ktorého bodu na x-ovej osi bude zobrazený výsledný graf, pr.: `x_from = -12.0`, implicitná hodnota je -5,
- `x_to` – nepovinný parameter, jeho hodnota zodpovedá, po ktorý bod x-ovej osi bude zobrazený výsledný graf, pr.: `x_to = 12.0`, implicitná hodnota je 5,
- `y_from` – nepovinný parameter, jeho hodnota zodpovedá, od ktorého bodu na y-ovej osi bude zobrazený výsledný graf, pr.: `y_from = -5.0`, implicitná hodnota je -5,
- `y_to` – nepovinný parameter, jeho hodnota zodpovedá, po ktorý bod na y-ovej osi bude zobrazený výsledný graf zobrazený, pr.: `y_to = 5.0`, implicitná hodnota je 5.

Nepovinné parametre pre manipuláciu s GA očakávané v sekcii **genetic**:

- `popsize` – parameter určujúci počet jedincov v populácií GA, implicitná hodnota parametru je nastavená na hodnotu 1000, maximálna hodnota tohto parametru je

10 000 a minimálna 10 jedincov, platí čím väčšia populácia tým dlhší výpočet ako aj presnosť výpočtu, pr.: `popsize = 290`, implicitná hodnota je 1000,

- `ngens` – parameter určujúci počet generácií GA, teda počet iterácií, ktoré sa v rámci výpočtu vykonajú, implicitná hodnota je 300, maximálna hodnota tohto parametru je 1000 a minimálna 10 generácií, pr.: `ngens = 200`, implicitná hodnota je 300,

- `pmut` – pravdepodobnosť že na jedincovi dôjde ku mutácií, implicitná hodnota je 0.01 (1%), minimálna hodnota je 0.0 maximálna 0.3, pr.: `pmut = 0,02`, implicitná hodnota je 0.01,

- `pcross` – pravdepodobnosť kríženia jedincov, implicitná hodnota je 0.9 (90%), minimálna hodnota je 0.0 maximálna 1.0, pr.: `pcross = 0,7`, implicitná hodnota je 0.9.

Sekcie pre jednotlivé intervaly očakávajú zhodné parametre. Parametre očakávane v sekcii **section_one**, **section_two**, **section_three**:

- `point` – viacnásobný povinný parameter, môže byť v rámci sekcie zadaný toľko krát, koľko bodov sa v danom intervale nachádza, nová hodnota musí byť uvedená kľúčovým slovom *point*, vždy na novom riadku, príklad viď koniec podkapitoly,

- `function` – voľba funkcie, ktorou budú zadané body aproximované, táto voľba sa dá samozrejme previesť aj za behu programu na panely užívateľskej interakcie pomocou roletového menu a výberu funkcie, zoznam funkcií a im odpovedajúce číselné kódy pre konfiguračný súbor sú uvedené v časti A.4.4, pr.: `function = 2`, implicitne je nastavená hodnota na 1.

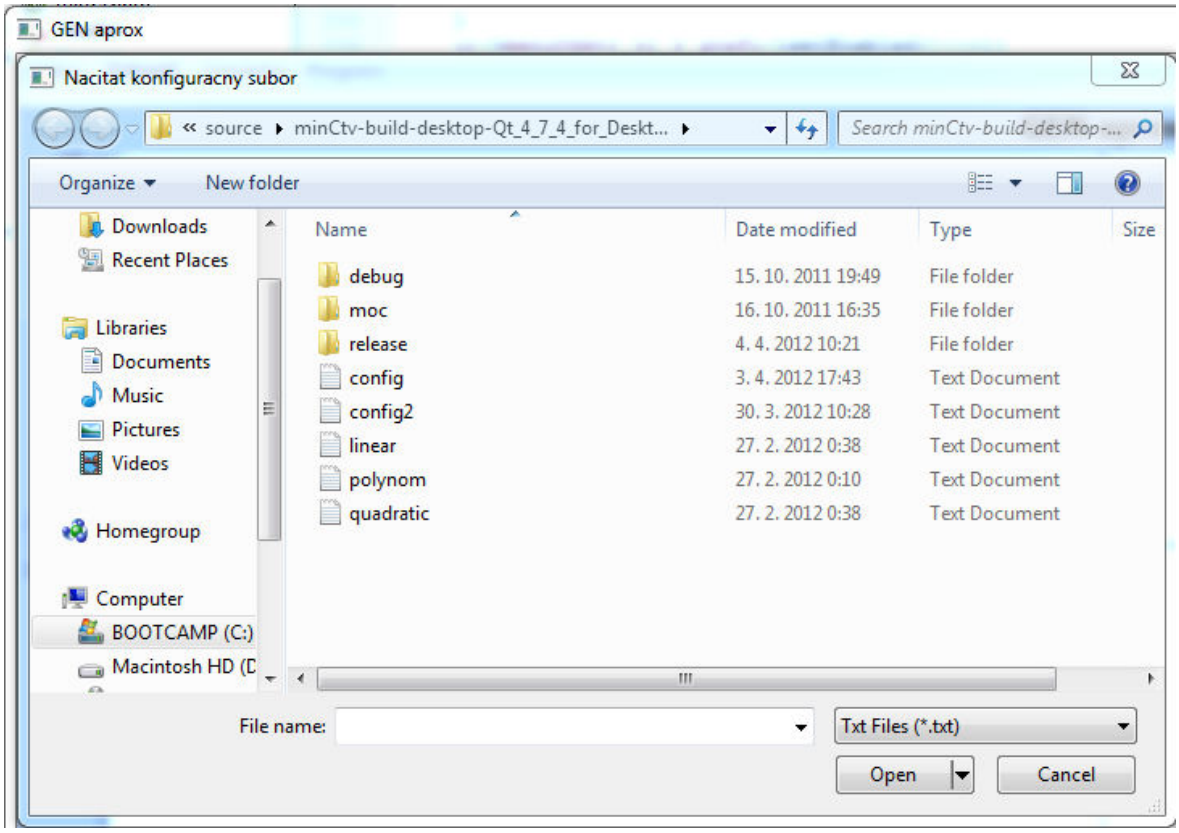
Príklad konfiguračného súboru

Nasledujúci konfiguračný súbor je priložený v archíve s aplikáciou.

```
[section_one]

point = [-5,0;-2.2]
point = [-4,0;-2.425]
point = [-3,0;-2.8]
point = [-2,0;-3.55]
point = [0,0;-5.8]
function = 2

[section_two]

point = [1.0;2.2]
point = [1.5;6.5]
point = [2.8;11.3]
point = [4.4;12.0]
point = [6.0;14.7]
point = [7.6;15.9]
point = [8.2;16.3]
point = [10.0;19.2]

[section_three]

point = [11.2;22.0]
point = [11.8;13.9]


[general]

int1 = -10.0
int2 = 0.5
int3 = 11.0
int4 = 12.0

x_from = -12
x_to = 12
y_from = -10
y_to = 10

[genetic]
pcross = 0.8
pmut = 0.02
popsize = 1100
ngens = 320
```

Načítanie konfiguračného súboru

V programe je možne konfiguračný súbor načítať dvoma spôsobmi. Prvý je pomocou klávesovej skratky CTRL+K, druhý cez hlavné menu Súbor – Načítať konfig. súbor. Po voľbe jednej z dvoch možností sa zobrazí nasledujúci dialóg zobrazený na obrázku A.1.



Obrázok A.1:
Dialóg na otvorenie konfiguračného súboru

V prípade chyby v konfiguračnom súbore je táto chyba vyhodnotená a oznámená chybovým hlásením.

A.4.2 Spustenie výpočtu

Výpočet sa spustí prevedením nasledujúcej série úkonov:

1. Korektné načítanie konfiguračného súboru. Pri akejkoľvek chybe, program túto chybu užívateľovi ohlási.
2. Nastavenie parametrov výpočtu:
    a. voľba aproximačných funkcií pre jednotlivé intervaly – na obrázku A.2 v červenom rámčeku,
    b. voľba aproximačnej metódy, buď metóda najmenších vzdialeností, alebo metóda najmenších štvorcov – obrázok A.2 v modrom ohraničení.

3. Spustenie výpočtu pomocou tlačítka Aproximuj – obrázok A.2 v zelenom rámčeku.

4. Ukončenie výpočtu nastane po zobrazení výsledných kriviek do plochy grafu a zároveň analytického predpisu týchto kriviek, zobrazených pod výberom aproximačných funkcií.

5. Vlastnosťou GA je dosiahnutie výsledkou s rôznou odlišnosťou od optima v rôznych behoch. Preto pokiaľ nie je užívateľ s nájdeným výsledkom spokojný, je možné previesť opakovaný výpočet pre načítaný konfiguračný súbor pomocou klávesovej skratky CTRL+N alebo pomocou menu Súbor -> Nový výpočet.

6. Pre výpočet s novými údajmi je potrebné načítať nový konfiguračný súbor.

Obrázok A.2:
Ovládanie spustenia výpočtu

A.4.3 Užívateľské akcie

Táto časť popisuje akcie, ktoré je možné s programom prevádzať. Tieto jednotlivé akcie sú mapované položkami hlavného menu nasledovne:

- Súbor:
    - Načítať konfig. súbor (CTRL+K) – otvorí dialóg na načítanie konfiguračného súboru.
    - Nový výpočet (CTRL+N) – pokiaľ už výpočet prebehol, odstráni aproximované krivky a ich analytické predpisy a je možné previesť výpočet znovu

pre rovnaký konfiguračný súbor, pre kompletne nový výpočet s novými parametrami je nutné načítať nový konfiguračný súbor.

- Koniec (CTRL+Q) – ukončí program.

- Zobraziť:
  - Krivky (CTRL+R) – zaškrtávacia voľba, na výslednom grafe zobrazí/odstráni výsledné krivky.
  - Body (CTRL+B) - zaškrtávacia voľba, na výslednom grafe zobrazí/odstráni vstupné body.
  - Intervaly (CTRL+I) - zaškrtávacia voľba, na výslednom grafe zobrazí/odstráni priamky určujúce hranice intervalov.
  - Všetky objekty (CTRL+O) – zaškrtávacia voľba, na výslednom grafe zobrazí/odstráni všetky vyššie menované objekty.

- Schránka:
  - Vzorec:
    - Interval 1 (CTRL+1) – uloží do schránky analytický predpis výslednej funkcie pre prvý interval.
    - Interval 2 (CTRL+2) – uloží do schránky analytický predpis výslednej funkcie pre druhý interval.
    - Interval 3 (CTRL+3) – uloží do schránky analytický predpis výslednej funkcie pre tretí interval.
    - Všetky intervaly (CTRL+4) – uloží do schránky analytický predpis výsledných funkcií pre všetky intervaly oddelené novým riadkom.
  - Graf (CTRL+G) – uloží do schránky výsledný graf.

- Program:
  - O programe (CTRL+A) – zobrazí základné informácie o programe.
  - Pomoc (CTRL+H) – zobrazí základy používania programu.

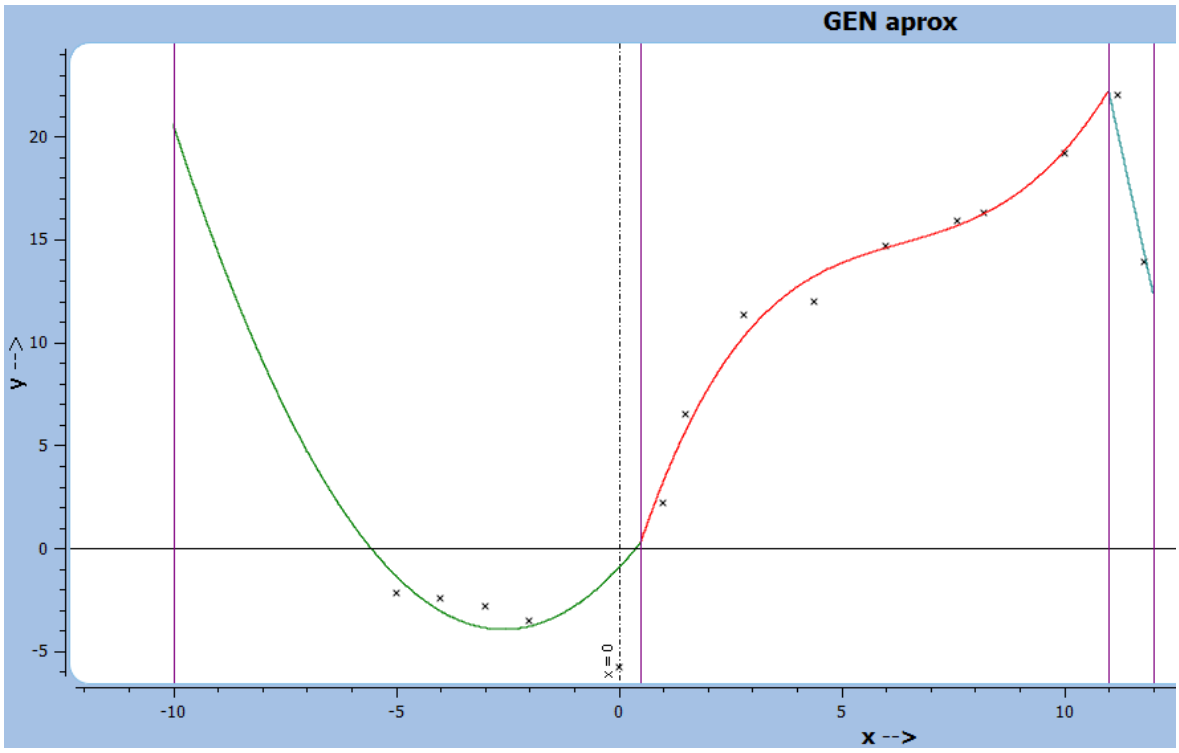
A.4.4 Zoznam podporovaných matematických funkcií

Programom GEN aprox je možné prekladať zadané vstupné body týmito matematickými funkciami:

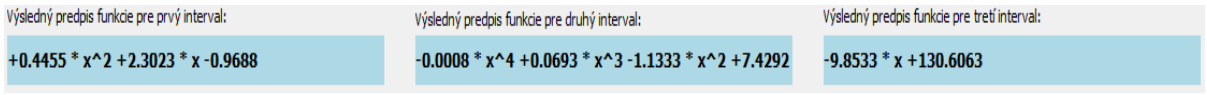
- lineárna funkcia: $y = ax + b$, voľba v konfiguračnom súbore: 1,
- kvadratická funkcia: $y = ax^2 + bx + c$, voľba v konfiguračnom súbore: 2,
- hyperbolická funkcia (1): $y = \frac{a}{x} + b$, voľba v konfiguračnom súbore: 3,
- hyperbolická funkcia (2): $y = \frac{a}{x^2} + bx + c$, voľba v konfiguračnom súbore: 4,
- polynomiálna funkcia tretieho stupňa: $y = ax^3 + bx^2 + cx + d$, voľba v konfiguračnom súbore: 5,
- polynomiálna funkcia štvrtého stupňa: $y = ax^4 + bx^3 + cx^2 + dx + e$, voľba v konfiguračnom súbore: 6,
- exponenciálna funkcia (1): $y = ae^{-x} + b$, voľba v konfiguračnom súbore: 6,
- exponenciálna funkcia (2): $y = ae^{b(-x)} + c$, voľba v konfiguračnom súbore: 7,
- logaritmická funkcia: $y = a \ln x + b$, voľba v konfiguračnom súbore: 8,
- logaritmus sínusu: $y = a \ln(2 + \sin x) + b$, voľba v konfiguračnom súbore: 9,
- sínusoida: $y = a \sin(x + b) + c$, voľba v konfiguračnom súbore: 10,
- sínus a cosínus: $y = a \sin(x) + b \cos(x) + c$, voľba v konfiguračnom súbore: 11.
- exponenciála a logaritmus: $y = ae^x + b \ln x + c$, voľba v konfiguračnom súbore: 12.
- sínus, cosínus a logaritmus: : $y = a \sin(x) + b \cos(x) + c \ln x + d$, voľba v konfiguračnom súbore: 13.

A.4.5 Výstup programu

Grafický výstup programu je tvorený výsledným grafom, zobrazeným na obrázku A.3 a analytickým predpisom výsledných funkcií, tak ako ukazuje obrázok A.4.



Obrázok A.3:
Grafický výstup programu - graf



Obrázok A.4:
Grafický výstup programu – analytický predpis

Príloha B: Aplikácia

Zdrojové kódy aplikácie v jazyku C++, výsledná preložená a spustiteľná aplikácia ako aj archív určený na distribúciu cez web sú umiestnené na priloženom CD.