VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

KLASIFIKACE POHYBOVÝCH ABNORMALIT POMOCÍ GENETICKÉHO PROGRAMOVÁNÍ

MOVEMENT ABNORMALITIES CLASSIFICATION USING GENETIC PROGRAMMING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ALEŠ CHUDÁREK**
AUTHOR

VEDOUCÍ PRÁCE **MICHAELA DRAHOŠOVÁ, Ing., Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Chudárek Aleš**
Program: Informační technologie
Název: **Klasifikace pohybových abnormalit pomocí genetického programování**
Movement Abnormalities Classification using Genetic Programming
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s principy evolučních algoritmů, genetického programování a klasifikace.
2. Navrhněte program umožňující provádět klasifikaci pomocí genetického programování.
3. Program z bodu 2 implementujte.
4. Ověřte funkčnost programu na zadaných úlohách z oblasti klasifikace pohybových abnormalit.
5. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucí práce.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Drahošová Michaela, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Při potlačování příznaků Parkinsonovy nemoci je pro pacienta velice důležité správné dávkovaní léků. Nesprávné dávkování může zapříčinit buďto nedostatečné potlačení příznaků nebo naopak při vysokých dávkách dochází k vedlejším účinkům, například dyskinezii. Ta se projevuje nedobrovolným pohybem svalů. Tato práce se zabývá problematikou automatizované klasifikace dyskinezie z pohybových dat nasnímaných pomocí tříosého akcelerometru umístěného na těle pacienta. V této práci je klasifikátor dyskinezie automatizovaně navrhován pomocí Kartézského genetického programování. Navržený klasifikátor dosahuje velmi dobré kvality při klasifikaci závažné míry dyskinezie (AUC = 0,94), což je srovnatelný výsledek jako u technik prezentovaných v odborné literatuře.

Abstract

When suppressing the symptoms of Parkinson's disease, the correct dosage of drugs is critical for the patient. Improper dosing can either cause insufficient suppression of symptoms or, conversely, side effects, such as dyskinesia, occur with high doses. Dyskinesia is manifested by involuntary muscle movement. This work deals with the automated classification of dyskinesia from motion data recorded using a triaxial accelerometer located on the patient's body. In this work, the classifier of dyskinesia is automatically designed using Cartesian genetic programming. The designed classifier achieves very good quality of classification of severe dyskinesia (AUC = 0,94), which is a comparable result to the techniques presented in scientific literature.

Klíčová slova

Strojové učení, Parkinsonova nemoc, evoluční algoritmus, kartézské genetické programování, dyskinezie, klasifikátor, evoluce.

Keywords

Machine learning, Parkinson's disease, evolutionary algorithm, cartesian genetic programming, dyskinesia, classifier, evolution.

Citace

CHUDÁREK, Aleš. *Klasifikace pohybových abnormalit pomocí genetického programování.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Michaela Drahošová, Ing., Ph.D.

Klasifikace pohybových abnormalit pomocí genetického programování

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením paní doktorky Michaely Drahošové. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Aleš Chudárek
9. května 2021

Poděkování

Na tomhle místě bych rád poděkoval své vedoucí doktorce Michaele Drahošové za pravidelné konzultace k projektu, za cenné informace, které byly předány systematicky, včas a hlavně srozumitelně.

Obsah

Kapitola 1

Úvod

V dnešní době se počítačové techniky neustále vyvíjejí a tím stoupá i úroveň problémů, které můžeme řešit. S tímto nám často pomáhá umělá inteligence, která začíná být velice používána ve všech oblastech našeho života. Velkou inspirací pro nynější algoritmy je právě příroda a její způsob řešení problémů. Algoritmy získané např. pomocí genetického programování dokáží najít postup i v situacích, které by člověk nikdy nevymyslel.V této práci budě využito kartézského genetického programování pro výpomoc v lékařství při léčbě Parkinsonovy nemoci.

Obecný cíl této práce je schopnost vypomoct lékařům s určením úrovně dyskinezie u pacientů. Pro tento proces se prozatím využívá sada pozorování, kde doktor sleduje pohyby pacienta při daných fyzických testech i normálních každodenních aktivitách. V této práci je využito těchto naměřených dat. Pomocí kartézského genetického programování (CGP) můžeme experimentováním postupně implementovat algoritmus, který zvládá klasifikovat dané úrovně s určitou přesností. Touto prací bych tedy chtěl experimentálně ověřit, že stejných výsledků jde dosáhnout i pomocí standardního CGP.

Obecné teoretické informace potřebné k porozumění problematiky se přiblíží v kapitole 2, kde je blíže vysvětlená terminologie a principy využitých algoritmů. Následující kapitola 3 popisuje návrh programu, který je předmětem této práce. V kapitole 4 popisujeme jeho implementaci, vstupní a výstupní data, průběh implementace, problémy při řešení i případná další možná rozšíření. Kapitola 5 nám popisuje, jak se s implementovaným programem pracovalo, s jakými parametry je potřeba ho spouštět a jaké výsledky byly nalezeny.

Kapitola 2

Teoretický základ práce

V této kapitole bude blíže vysvětlen princip evolučních algoritmů, genetického programování a CGP. Jsou zde vysvětleny veškeré pojmy použity dále v práci včetně popisu klasifikátoru. Tyto informace byly přebrány z knihy Evoluční hardware [8].

2.1 Evoluční algoritmy

Evoluční algoritmy jsou prohledávací algoritmy založené na principu přirozené selekce, kterou Charles Darwin popisuje ve své knize O původu druhů [1]. Tato kniha popisuje způsob, jakým se vyvíjel život na zemi. Díky rozmnožování a předávání genetické informace nové generaci může dojít k její změně. Tato genetická úprava má pak vliv na schopnosti konkrétního jedince. Pokud tento jedinec není geneticky vhodně vybaven pro přežití v jeho prostředí, tak umírá. Na druhou stranu jedinci s pozitivní genetickou změnou přežijí déle, a proto je více pravděpodobné, že tuto genetickou změnu předají další generaci.

2.1.1 Pojmy

Jelikož je problematika velmi inspirována biologií, spousta pojmů je podobná, ba i stejná. Je tedy vhodné je definovat.

- **Gen** – Základní stavební jednotka chromozomu. Jedná se o hodnotu patřící do konečné množiny nad definovanou abecedou (binární čísla, celá čísla, atd.).
- **Chromozom** – Obvykle lineární pole genů. Reprezentuje jedno řešení ve stavovém prostoru.
- **Genotyp** – Kódovaný tvar řešení pomocí chromozomu.
- **Fenotyp** – Kandidátní řešení daného problému, které je sestaveno pomocí chromozomu. V některých evolučních algoritmech se jedná přímo o genotyp, jinde je potřeba mapovat.
- **Populace** – Množina kandidátních řešení, která má obvykle konstantní počet jedinců.
- **Fitness funkce** – Funkce, která určí vhodnost jedince. Na základě ohodnocení fitness lze určit, který z porovnávaných jedinců plní požadovanou funkci lépe.

2.1.2 Princip evolučního algoritmu

Evoluční algoritmy pracují podle zavedeného postupu, který je popsán níže.

- **Vytvoření počáteční populace** – První krok evolučního algoritmu je vytvoření počáteční populace. Jako populace je myšlena množina předem zvoleného počtu kandidátních řešení. Tato počáteční populace může být vytvořena buď náhodně nebo, pokud je třeba zdokonalit již známé řešení, je možné zakódovat toto řešení již do počáteční populace.

- **Ohodnocení populace** – Pro každou generaci je třeba ohodnotit její jedince. K tomu slouží fitness funkce. O funkci f obvykle nic nepředpokládáme, nemusí být tedy spojitá, mít derivaci nebo být úplně definovaná. Je pro nás černou skříňkou, která umí vrátit funkční hodnotu pro zadaný argument. Obvykle je ve formě $f : G \to \mathbb{R}$, kde G je množina všech možných genotypů [7]. Lze tedy určit, který ze dvou genotypů $g_1, g_2 \in G$ je vhodnějším řešením zadané úlohy porovnáním $f(g_1)$ a $f(g_2)$. Hodnota $f(g)$ pak označuje fitness hodnotu genotypu g. Příklad fitness funkce bude uveden níže v sekci 3.1.4.

- **Výběr rodičů (selekce)** – Jakmile jsou všichni jedinci ohodnoceni je třeba vybrat rodiče, kteří budou zdrojem následující generace. Existuje několik způsobů, jak rodiče zvolit.

  - **Deterministická selekce** – Nejjednodušší varianta. Je vždy vybráno n jedinců s nejvyšším fitness ohodnocením.
  - **Turnajová selekce** – Pro každého požadovaného rodiče je vytvořen turnaj. Je vybráno n jedinců, kteří jsou po dvojici srovnáváni a posíláni do dalšího kola, dokud není nalezen vítěz, který se pak stává rodičem.
  - **Proporcionální selekce** – Definuje pravděpodobnost, že bude jedinec vybrán jako rodič. Tato pravděpodobnost je přímo úměrná absolutní hodnotě fitness jedince. Je využito algoritmu tzv. kola štěstí, kdy je každému jedinci přiřazen podinterval z intervalu <0,1> ve stejném poměru, jako je poměr jejich absolutních hodnot fitness. Rodič je pak vybrán pomocí náhodného čísla z intervalu <0,1>. Podinterval, kterému toto náhodné číslo paří pak určuje rodiče.

- **Nová generace** – K vytvoření nové generace jsou potřeba rodiče. Chromozom rodičů je pak přenesen na jejich potomky za použití genetických operátorů. Nejznámějšími genetickými operátory jsou křížení a mutace.

  - **Křížení** – Při křížení dochází k výměně části chromozomu mezi rodiči.
  - **Mutace** – Při vytváření nových potomků může dojít ke změně některého z genů, který je nahrazen genem náhodným. Mutace probíhá nedeterministicky. Pravděpodobnost mutace je předem určena uživatelem.

  Existují dvě varianty tvorby nové generace.

  - **Generační varianta** – Nová generace je složena pouze z nových potomků, tudíž se žádný z rodičů v nové generaci nevyskytne.
  - **Překrývání populace** – Nová generace je složena z kombinace jedinců z předchozí generace a potomků rodičů.

- **Ukončovací podmínka** – Algoritmus je ukončen poté, co nalezne dostatečně kvalitní kandidátní řešení nebo je dosaženo předem určeného počtu generací.

```
evoluani_algoritmus(){
    t = 0;
    P(t) = vytvor_pocatecni_populaci;
    ohodnot P(t);
    while (ukoncovaci_podminka == FALSE) do
    {
        Q(t) = vyber_rodice(P(t));
        Q'(t) = vytvor_nove_jedince(Q(t));
        ohodnot (Q'(t));
        P(t + 1) = vyber_jedince_do_nove_populace(P(t), Q'(t));
        t = t + 1;
    }
}
```

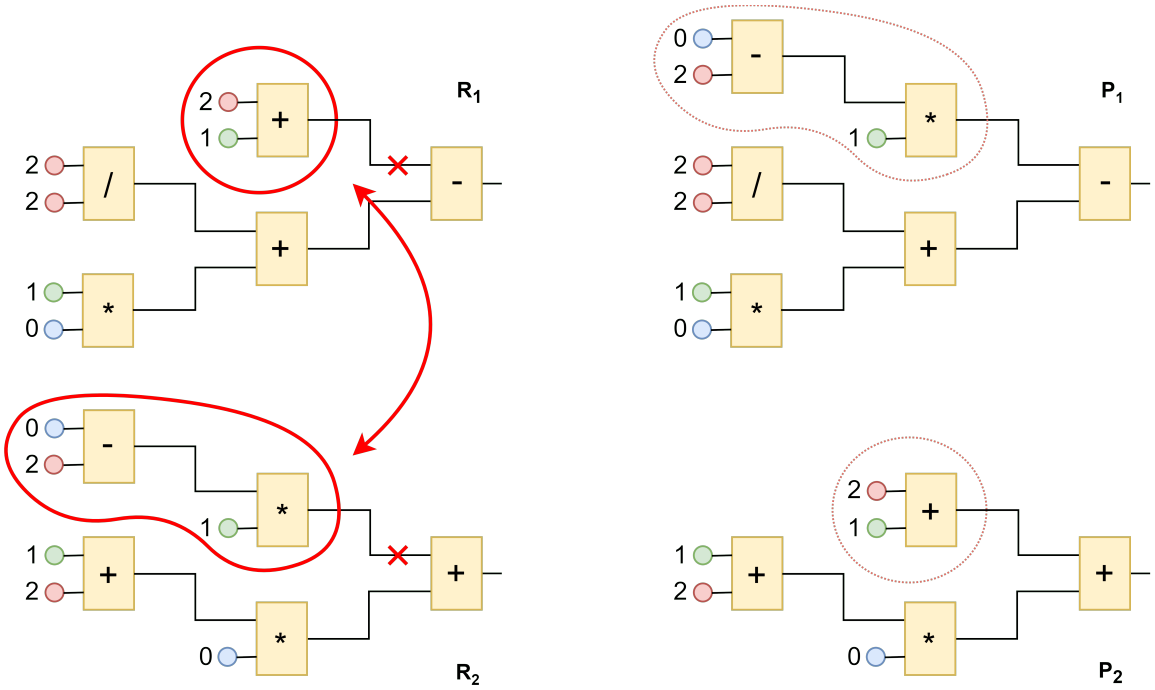
Obrázek 2.1: Popis evolučního algoritmu pomocí pseudokódu z knihy [8]

2.1.3 Genetické programování

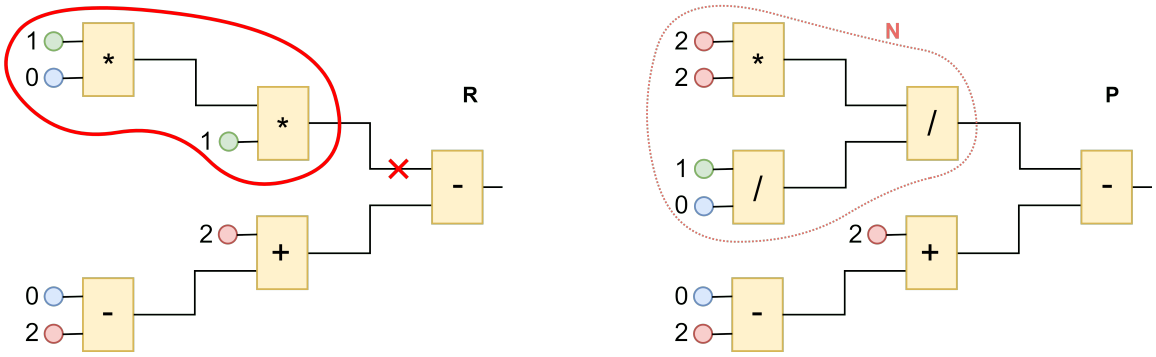
Jedná se o jednu z variant evolučního algoritmu, která vznikla koncem 80. let 20. století. Velkou zásluhu na jeho vývoji si zasloužil John Koza. Díky genetickému programování se již nehledají pouze optimální hodnoty parametrů v chromozomu, ale lze generovat již celé programy. Algoritmus genetického programování je velice podobný obecnému evolučnímu algoritmu. Využívá se jak generační varianta, tak překrývání populace. V čem se liší od obecného evolučního algoritmu je popsáno dále.

- **Reprezentace** – Genetické programování pracuje se spustitelnými strukturami. Jedná se například o programy reprezentovány stromy.
- **Genetické operátory** – Kromě křížení a mutace existuje řada operátorů umožňujících např. evolučně vytvářet podprogramy v rámci vyvíjených programů.
- **Vyhodnocení fitness** – Kandidátní program je spuštěn nad vstupními daty a výstupy kandidátního programu jsou porovnány s výstupy, které požadujeme od plně funkčního řešení
- **Množina terminálů**–Terminály jsou myšleny listové uzly v chromozomu. Jedná se o funkce bez argumentů nebo konstanty z podmnožiny reálných čísel.
- **Množina funkcí** – Použité funkce se mohou lišit podle oblasti ve které se pohybujeme. Je potřeba zajistit, aby všechny možné výstupy jednotlivých funkcí mohly být zároveň použity jako argumenty k těmto funkcím. Tím je zajištěna funkčnost napojování funkcí do stromu. Z toho důvodu se používají takzvané chráněné varianty těchto funkcí, které dokáží zpracovat i okrajové situace jako je např. dělení nulou nebo odmocnina ze záporného čísla. Výstupem této funkce pak může být konstanta nebo aproximovaná hodnota. Množina funkcí tedy obvykle obsahuje standardní aritmetické a logické funkce.

- **Reprezentace programů** – Programem jsou myšleny struktury sestávající z funkcí a terminálů. Dále pravidla používání těchto funkcí a terminálů například struktura stromu, lineární struktura, obecné grafové struktury, apod.
- **Operátory** – Genetické operátory fungují stejně jako u evolučního algoritmu. Základem je opět křížení a mutace.
  - **Křížení** – Existuje řada způsobů. U stromové struktury se často používá způsob záměny části chromozomu. Pro dva vybrané jedince (rodiče) se náhodně zvolí uzel, jejichž podstromy jsou následně prohozeny. Tímto vznikají noví dva potomci. Příklad křížení můžeme vidět na obrázku 2.2.
  - **Mutace** – Při mutaci je pracováno pouze s jedním jedincem, u kterého s jistou pravděpodobností může dojít k mutaci. Pro stromovou strukturu pak platí, že se náhodně vybere uzel, jeho podstrom je nahrazen za náhodný podstrom, generovaný stejným způsobem, jako při generování původní populace. Příklad mutace můžeme vidět na obrázku 2.3.
- **Fitness funkce** – Závisí na dané oblasti. Nejčastěji se kandidátní program trénuje na trénovací množině. Jedná se o množinu vstupů s očekávaným vyhodnocením. Výstupy tohoto programu jsou pak porovnávány s očekávaným výsledkem. Fitness pak může být například průměrná odchylka od daných hodnot. Tuto odchylku se pak snažíme minimalizovat. Po skončení evoluce je program testován na testovací množině. Cílem je najít program, který co nejlépe aproximuje zadaná data.



Obrázek 2.2: **Křížení** – U rodičů R_1 a R_2 jsou vybrána místa pro křížení. Z těchto míst se vezmou celé podstromy, které se navzájem prohodí. Tím vznikají noví dva potomci P_1 a P_2.



Obrázek 2.3: **Mutace** – U jednoho rodiče R se vybere místo pro mutaci. Celý podstrom z tohoto místa je nahrazen náhodným podstromem N. Tím vzniká jeden potomek P.

2.1.4 Kartézské genetické programování (CGP)

Kartézské genetické programování (dále jen CGP) je jedna z variant genetického programování, kde jednotlivá kandidátní řešení jsou zakódovaná jako orientované acyklické grafy. Graf je zde reprezentován jako dvojrozměrná mřížka uzlů. Každý z těchto uzlů pak nese dvě informace. Operaci, kterou má uzel vykonat a adresy všech jeho vstupů. Princip CGP byl převzat z knihy [5].

Při práci s CGP je potřeba specifikovat následující parametry algoritmu.

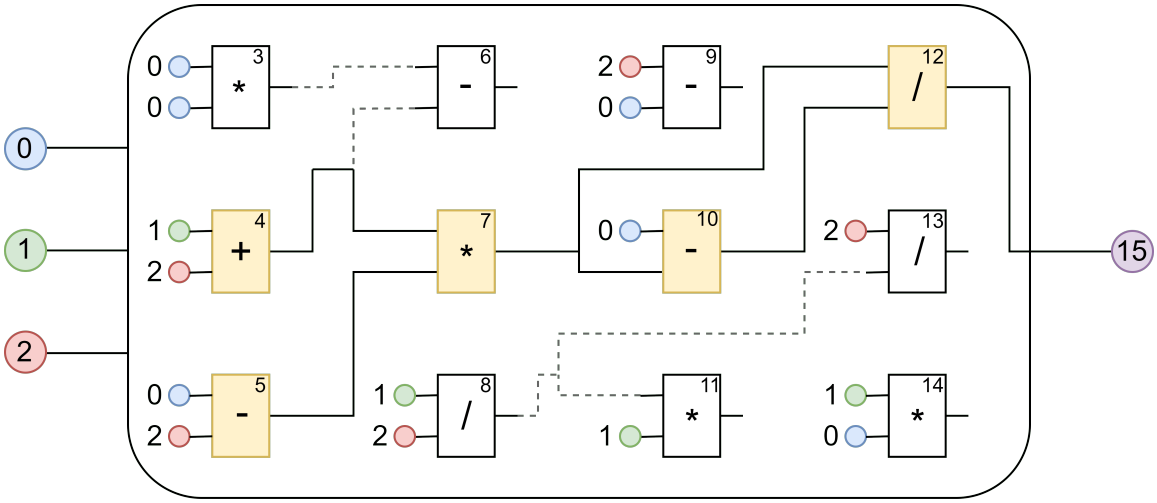
Parametry

- $\mathbf{n_c}$ – Počet sloupců v dvojrozměrné mřížce uzlů.
- $\mathbf{n_r}$ – Počet řádků v dvojrozměrné mřížce uzlů.
- $\mathbf{n_i}$ – Počet primárních vstupů kandidátního řešení.
- $\mathbf{n_o}$ – Počet primárních výstupů kandidátního řešení.
- $\mathbf{n_n}$ – Maximální počet argumentů pro zvolené funkce.
- $\mathbf{\Gamma}$ – Množina zvolených funkcí
- $\mathbf{n_f}$ – Počet zvolených funkcí. $n_f = |\Gamma|$
- **L (L-back)** – Určuje míru propojitelnosti

Na obrázku 2.4 můžeme vidět jedno možné kandidátní řešení CGP. Pro tohle konkrétní řešení byly zvoleny parametry $n_c = 4$, $n_r = 3$, $n_i = 3$, $n_o = 1$, $n_n = 2$, $\Gamma = \{+ \to 0, - \to 1, * \to 2, / \to 3\}$, $n_f = 4$, $L = 2$.

Popis chromozomu

Samotný jedinec se tedy vytváří pomocí kombinace těchto parametrů. Počet primárních vstupů pro daný chromozom je určen typem úlohy. Stejně tak i počet jeho primárních výstupů. Velikost mřížky uzlů značně určuje složitost řešení. Pro příliš velkou mřížku může prohledávání trvat zbytečně dlouho, tím pádem nemusíme včas nalézt požadované kandidátní řešení. Na druhou stranu příliš malá mřížka nemusí umožňovat dostatečný počet



Obrázek 2.4: Příklad kandidátního řešení CGP.

operací pro vyřešení úlohy. Ideální rozměry mřížky jsou proto potřeba nalézt experimentálně.

Úzce spojený s počtem sloupců je také parametr L-back. Ten nám určuje jaké vstupy může daný uzel mít. O situaci, kdy je L-back = 1, říkáme, že je propojitelnost nejmenší. To znamená, že každý uzel může mít jako vstupy buď vstupy primární a nebo výstupy z uzlů v předchozím sloupci. Jestliže se L-back = n_c, můžeme uzel propojit s jakýmkoli uzlem z předcházejících sloupců. Na obrázku 2.4 můžeme vidět, že uzel 12 používá jako jeden ze vstupů výstup uzlu 7, který je vzdálený o 2 sloupce.

Množina funkcí záleží opět na typu úlohy. Pro řešení úloh, kde je výsledkem aritmetická funkce, používáme jednoduché aritmetické funkce (+, -, *, /, MIN, MAX, AVG, sin(x), ...). Pro řešení úloh, kde je výsledkem logická operace, používáme logické funkce (AND, OR, NOT, XOR, ...). Tyto funkce nám určují maximální počet možných vstupů (n_n).

Reprezentace chromozomu

Pro popis chromozomu používáme Λ_{CGP} celých čísel, které popisují jeho rozměry. Hodnotu Λ_{CGP} získáme tímto vzorcem $\Lambda_{CGP} = n_r n_c(n_n + 1) + n_o$. Při kódování pak postupujeme takto: Všechny primární vstupy jsou popsány indexy 0, ..., $n_i - 1$. Dále jsou indexovány jednotlivé uzly. Indexuje se postupně po sloupcích. První uzel v prvním sloupci pak má index n_i a poslední uzel diagonálně na druhé straně tabulky má index $n_i + n_c n_r - 1$. Další v pořadí indexování jsou primární výstupy. První výstup má tedy index $n_i + n_c n_r$ a poslední $n_i + n_c n_r + n_o - 1$.

Inicializace náhodného chromozomu

Pro vytvoření již konkrétního chromozomu je zapotřebí kromě parametrů zvolit jednotlivé funkce uzlů a vše důkladně propojit.

Každý uzel je reprezentován k-ticí celočíselných hodnot, kde $k = n_n + 1$. Prvních n_n hodnot určuje indexy uzlů, které budou použity jako argumenty do funkce. Poslední hodnota pak reprezentuje index funkce tohoto uzlu. Pokud se podíváme na obrázek 2.4, můžeme říci, že například uzel s indexem 12 můžeme zapsat takto: (7, 10, 3). Čísla 7 a 10 reprezen-

tují adresy dvou argumentů a číslo 3 reprezentuje index funkce, v tomto případě se jedná o dělení. Celý chromozom na obrázku 2.4 bude vypadat takto:

0,0,2; 1,2,0; 0,2,1; 3,4,1; 4,5,2; 1,2,3; 2,0,1; 0,7,1; 8,1,2; 7,10,3; 2,8,3; 1,0,2; 15

Počáteční populace je často generována náhodně, avšak v povoleném rozsahu.

Pro náhodné zvolení operace uzlu stačí náhodně vybrat číslo z množiny $\{0, 1, ..., n_f - 1\}$.

Pokud chceme vygenerovat platnou hodnotu i pro vstupní argument uzlu C_x, kde x je index tohoto uzlu, využíváme množiny Υ_x, která obsahuje indexy všech možných vstupů. Tím jsou myšleny všechny primární vstupy nebo uzly z L předchozích sloupců. Platí tedy $i \in \Upsilon_x$. Maximální počet možných vstupů $|\Upsilon_x|_{MAX}$ pro daný uzel C_x, vychází ze vzorce

$$|\Upsilon_x|_{MAX} = n_i + L \cdot n_r. \tag{2.1}$$

Množina Υ_x tedy vždy obsahuje hodnoty $0, ..., n_i - 1$ zastupující primární vstupy. Pro hodnoty zastupující vhodné uzly budeme postupovat následovně. Nejprve musíme zjistit index sloupce uzlu C_x (sloupce jsou indexovány od nuly zleva doprava). Tento sloupec označíme c. Pro výpočet c pomocí indexu x použijeme vzorec

$$c = \frac{(x - n_i) - (x - n_i) \mod n_r}{n_r}. \tag{2.2}$$

Pokud nám vyjde $c \leq L$, pak se nacházíme v takovém sloupci, že můžeme vybírat jako argumenty jakékoli uzly ze sloupců $< c$. Proto kromě primárních vstupů prvky $\{n_i, n_i + 1, ..., x - (x - n_i) \mod n_r - 1\} \in \Upsilon$.

Pokud je $c > L$, tak do množiny Υ patří pouze prvky ze sloupců $c - L$ až $c - 1$. Tudíž $\{x - L * n_r - (x - n_i) \mod n_r, ..., x - (x - n_i) \mod n_r - 1\} \in \Upsilon$.

Kromě samotných uzlů je třeba ještě určit, kam se odkazují primární výstupy chromozomu. Primární výstupy mohou být napojeny na kterýkoliv primární vstup nebo výstup jakéhokoli uzlu. Proto stačí náhodně vygenerovat hodnotu z množiny $\{0, 1, ..., n_r n_c + n_i - 1\}$

Vlastnosti CGP

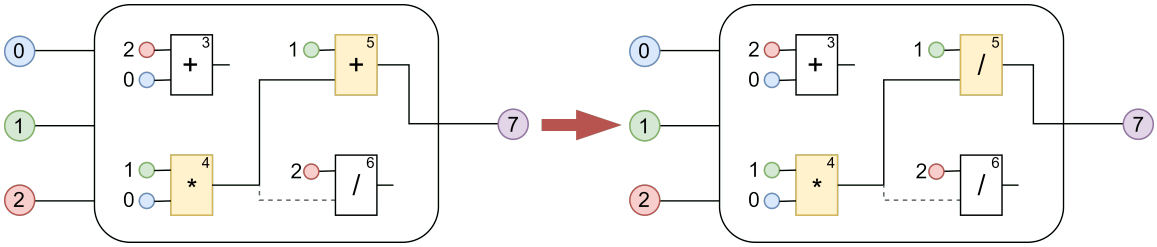
Základní vlastností uvedeného kódování je, že zatímco velikost chromozomu je pro celou populaci stejná, výsledná složitost zakódovaného obvodu je variabilní. Vzniká tak několik úrovní redundance.

- Ne každý uzel musí být použit, naopak některé uzly mohou být použity i vícekrát.
- Pokud máme funkce s počtem vstupů $< n_f$, zbylé vstupy pak nejsou využity.
- Některé z primárních vstupů nejsou využity.

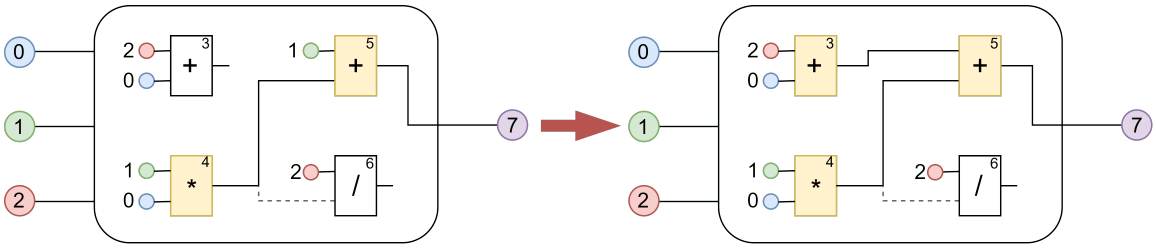
Uzly, které v chromozomu nejsou využity nazýváme neaktivní. Užité uzly jsou pak aktivní. Tento stav uzlu se může měnit pomocí genetických operátorů..

Genetické operátory - mutace

Standardní varianta CGP nepoužívá křížení pouze mutace. Mutací lze změnit jak funkce uzlů, tak i její argumenty. Oba případy můžeme vidět na obrázcích 2.5 a 2.6. Počet mutací jedince je argument určený při spouštění evoluce. Určuje, kolik genů z chromozomu bude při mutaci změněno.

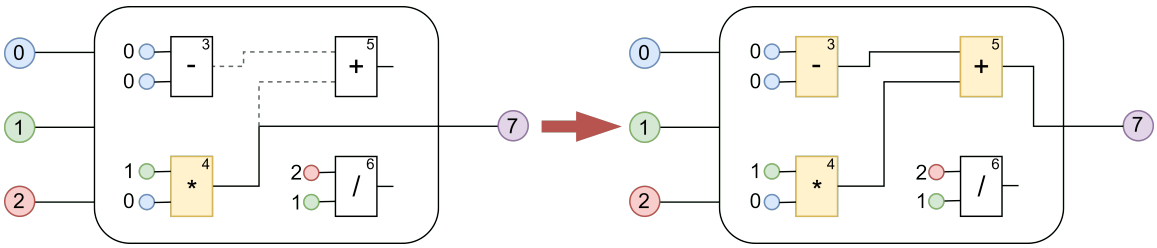


Obrázek 2.5: Ukázka mutace funkce uzlu CGP chromozomu. Uzel 5 změnil funkci + (0) na / (3).

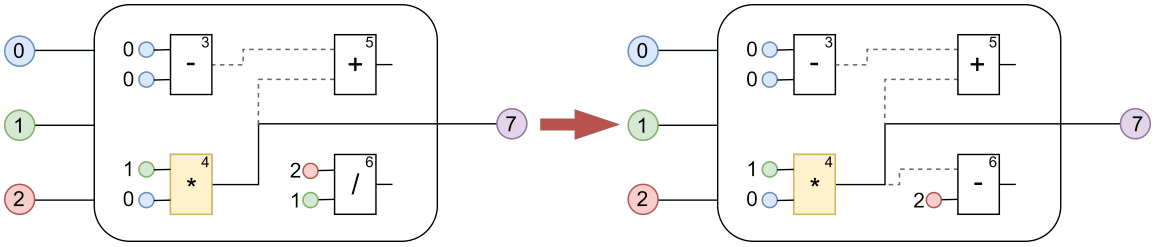


Obrázek 2.6: Ukázka mutace napojení uzlu CGP chromozomu. Uzel 5 změnil první argument z 1 na 3.

- **Neutrální mutace** – Pokud nastane mutace, která nezměnila hodnotu fitness jedince, nazýváme tuto mutaci jako neutrální. Můžeme toho dosáhnout dvěma způsoby. Buď nastane situace, kdy je fenotyp změněn, ale hodnotu fitness tato změna nepostihla (například přičtení nuly k výsledné fitness). Tuto variantu neutrální mutace můžeme vidět v obrázku 2.7, kde primární výstup po mutaci odkazuje na uzel 5, který změní fenotyp, ale hodnota fitness zůstane v tomto případě stejná. Druhý způsob je, že je mutace aplikována na neaktivní uzly. Na obrázku 2.8 můžeme vidět, že neaktivní uzel 6 zůstal neaktivní, ale zmutoval si jak vstupy, tak funkci. Obě tyto mutace nemají vliv na výslednou fitness hodnotu, jedná se tedy také o mutaci neutrální.
- **Adaptivní mutace** – Pokud není mutace neutrální, nazýváme ji adaptivní. Tato mutace má již vliv na změnu fitness funkce. Pokud vznikne řada neutrálních mutací následovaná jednou adaptivní mutací, která aktivuje některé z doposud neaktivních uzlů, může dojít k velké změně fenotypu. Mutace tímto způsobem neposkytuje jen malé úpravy fenotypu, ale také velké změny. Tím nahrazuje i explorativní operátor, kterým typicky bývá křížení. Z toho důvodu nám stačí operátor mutace.



Obrázek 2.7: Ukázka neutrální mutace CGP chromozomu se změnou ve fenotypu.



Obrázek 2.8: Ukázka neutrální mutace CGP chromozomu bez změny ve fenotypu.

Vytvoření nové generace

Při evoluci se do nové generace posune jeden jedinec (rodič) z poslední generace, který byl ohodnocen nejvyšší hodnotou fitness. Tento jediný rodič je mutován na další (většinou 4) potomky. Pokud má při výběru rodiče více jedinců stejné ohodnocení, bere se vždy ten, co nebyl rodičem v minulé generaci. Tím se zajistí genetická diverzita. Jedná se tedy o variantu překrývání populace. Přesněji jde o variantu, kde má nová generace $1 + \lambda$ jedinců. Jde o jednoho rodiče a λ potomků.

Algoritmus CGP

1. Vygenerování počáteční generace. Náhodně nebo s použitím již existujících řešení.
2. Ohodnocení celé populace pomocí fitness funkce.
3. Nalezení nejlépe ohodnoceného jedince v populaci.
4. Z vybraného rodiče vygenerujeme pomocí mutace λ potomků.
5. Tento rodič společně se svými λ potomky tvoří novou generaci.
6. Pokud není splněna ukončovací podmínka, pokračuje se krokem 2.

Vlastnosti CGP

Jelikož napojení jednotlivých uzlů je, alespoň na začátku, velice chaotické, může nastat situace, kdy některý uzel není v chromozomu vůbec využit (uzly 3, 6, 8, 9, 11, 13 a 14 v chromozomu na obrázku 2.4). Tyto uzly označujeme za neaktivní. Uzly využity ve výpočtu jsou pak uzly aktivní. Díky mutaci tak můžou jednotlivé uzly měnit, zda jsou aktivní, či nikoli.

2.2 Klasifikátor dyskinezie

V této práci je využito kartézského genetického programování k vytvoření klasifikátoru dyskinezie při léčbě Parkinsonovy choroby.

Parkinsonova choroba Parkinsonova choroba je nevyléčitelné neurologické onemocnění, které postihuje 1-2 lidi z tisíce po celém světě. Nejčastěji se choroba projevuje po 50. roku života, ale 10-15% vzplanutí tohoto onemocnění se objeví již před čtyřicátým rokem života. Při tomto onemocnění dochází k úbytku nervových buněk, díky kterým naše tělo produkuje dopamin. Dopamin umožňuje přenos signálů mezi nervovými buňkami. Onemocnění

se u pacientů může projevit únavou, minimální mimikou, pomalými nekoordinovanými pohyby, obtížnou artikulací, ale nejznámějším projevem je nedobrovolný třes.

Dyskinezie Jelikož je Parkinsonova nemoc zatím neléčitelná, existuje množství léků (například Levodopa), které potlačují její příznaky. Tyto léky pak nahrazují nedostatek dopaminu v těle. Při dlouhodobém užívání těchto léků však může docházet k vedlejším účinkům. Mezi ně patří především nedobrovolné trhavé pohyby svalů po celém těle. Tomuto jevu říkáme dyskinezie. Dyskinezie je velice častá. U pacientů užívajících náhražek dopaminu bylo zjištěno, že přes 90% případů trpí určitou mírou dyskinezie.

Pro snížení rizika dyskinezie je tedy potřeba správně dávkovat léky a včas snížit dávkování, pokud se dyskinezie objeví. Problém nastává ovšem v tom, že pacienti často sami neví, že trpí dyskinezií. Jelikož pravidelné prohlídky nejsou tak časté a ne vždy se podaří dyskinezii zachytit bylo vyvinuto zařízení pro jednoduché měření pohybových aktivit pacientů z pohodlí domova.

Snímající zařízení Pro jednoduché sbírání informací o pohybech pacienta bylo podle článku Michaela Lonese [4] využito 6 zařízení, které jsou rozmístěny po těle pacienta (hlava, hrudník, paže, stehna). Tato jednotlivá zařízení jsou vybavena tříosým akcelerometrem a tříosým gyroskopem, které pracují s frekvencí 100Hz. Tato data by pak měla být bezdrátově posílána do mobilního zařízení, které pomocí vhodného klasifikátoru správně vyhodnotí pacientův stav.

Obrázek 2.9: Senzory snímající akceleraci a rotaci částí pacientova těla. Jednotlivé senzory jsou umístěny na šesti místech (hlava, hrudník, paže, stehna).

Klasifikátor

Pro určování dyskinezie se využívá pětistupňového hodnocení (0-4), kde 0 reprezentuje pacienta, který je diagnostikován bez známek dyskinezie a 4 reprezentuje nejzávažnější dyskinezii. Při vytváření klasifikátoru tedy hledáme program, který zvládá co nejpřesněji přiřadit nasbíraná pohybová data do jedné z těchto pěti kategorií. Pro návrh klasifikátoru byla v této práci využita stejná data jako v článku od Michaela Lonese [4]. Tato data byla získána ze dvou studií popsaných na obrázku 2.10. Jako trénovací data byla pak použita data ze studie č. 1 a jako testovací data byla použita data ze studie č. 2. Ve článku Michaela

Lonese se můžeme dočíst, že pro návrh vhodného klasifikátoru zvolili upravenou variantu standardního CGP, tzv. implicitní kontextová reprezentace kartézského programování (dále jen **IRCGP**). IRCGP je verze CGP, kde se jako operátor kromě mutace objevuje také křížení.

	Study 1	Study 2
Number of patients	6	17
Assessment period (hours)	6	2
Gender male:female	4:2	11:6
Age (years)	71±8.9	65±7.3
Disease duration (years)	9.8±3.7	8.1±3.6
MDS-UPDRS motor score	31±19.1	28±18.0
PDYS-26 quality of life score	37.6±29.2	34.7±24.5

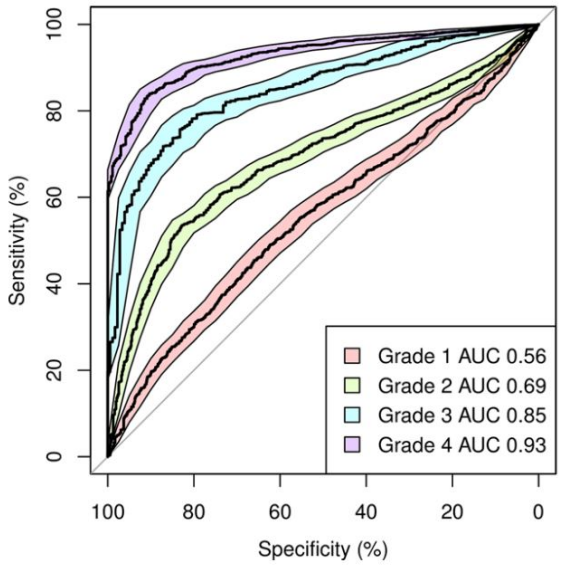
Obrázek 2.10: Popis dat ze dvou studií použitých pro trénování a testování klasifikátoru. Převzato z článku [4].

Model klasifikátoru

Model klasifikátoru ze článku [4], použitý k nalezení funkce, která kvalitně klasifikuje dyskinezii, využívá CGP chromozomu s 36 možnými operacemi v tabulce uzlů 6x6. Byla použita množina funkcí $\Gamma = \{+, -, *, /, \text{MEAN}, \text{MIN}, \text{MAX}, \text{AVG}\}$. Jako primární vstupy je bráno 32 hodnot z trénovacích dat ze studie č. 1 pomocí plovoucího okna. Tento postup je podrobněji popsán níže. Touto hodnotou je myšlena magnituda vypočítaná ze záznamů tříosého akcelerometru. Byly využity pouze záznamy, které prokazovaly vysokou míru dyskinezie (úroveň 3 a 4) a záznamy pacientů bez příznaků dyskinezie (úroveň 0). Nejlépe ohodnocený chromozom je pak otestován na druhé sadě dat ze studie č. 2. Testy jsou provedeny čtyři. Každý test spouštíme se záznamy jedné konkrétní úrovně dyskinezie a se záznamy úrovně 0. Jako fitness funkce byla použita AUC, která je podrobněji rozepsaná v sekci 3.1.4. Výsledné hodnoty klasifikátoru jsou zaznamenány v obrázku 2.11. Při porovnávání vysoké úrovně dyskinezie (4) se záznamy bez dyskinezie (0) zvládá tento klasifikátor rozlišovat mezi třídami s $AUC = 0,93$. S klesající úrovní dyskinezie klesá i schopnost klasifikátoru rozlišovat mezi třídami. Pro klasifikaci úrovně 3 dosahuje $AUC = 0,85$, pro úroveň 2 $AUC = 0,69$ a pro úroveň 1 $AUC = 0,56$.

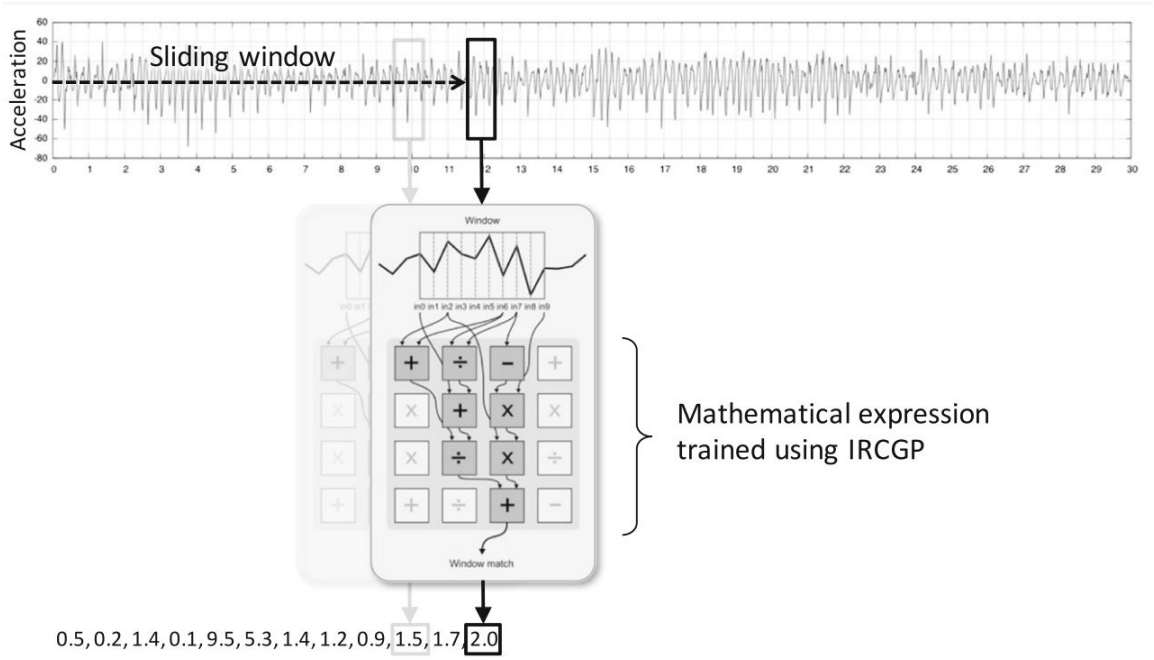
Načítání dat - Plovoucí okno

Jako vstupní data jsou brány csv záznamy tříosého akcelerometru s frekvencí záznamu 100Hz. Každý záznam tvoří tři hodnoty z tříosého akcelerometru a 3 hodnoty z tříosého gyroskopu. Každý záznam je potřeba převést na jedinou hodnotu. Jelikož je využito pouze akcelerometru, vypočte se ze záznamu magnituda. Pro ohodnocení jednoho souboru záznamů je využito plovoucího okna. Tento princip můžeme vidět na obrázku 2.12. To znamená, že je daný soubor dat ohodnocován zvlášť po 0,32 s. Pro první ohodnocení jsou jako primární vstupy použity magnitudy v čase 0 až 31, dále pak v čase 1 až 32, atd. Celkově je tedy pro jeden soubor o N záznamech generováno $N - 31$ ohodnocení. Celkové ohodnocení



Obrázek 2.11: Výsledné ohodnocení nejlepšího klasifikátoru ze článku Michaela Lonese [4].

tohoto souboru je pak aritmetický průměr těchto $N - 31$ hodnot. Pomocí vhodně zvolených prahů se určí míra dyskinezie.



Obrázek 2.12: Ukázka výběru primárních vstupů pomocí plovoucího okna. Přebráno ze článku Michaela Lonese [4].

Kapitola 3

Návrh

V této kapitole se budeme věnovat návrhu samotného klasifikátoru dyskinezie pomocí klasického CGP. Oproti článku Michaela Lonese [4] se tedy využije pouze operátoru mutace na rozdíl od IRCGP, kde bylo využito i křížení.

Jako trénovací a testovací data jsou poskytnuty csv soubory, které jsou popsány výše v sekci 2.2. Obsahují časové záznamy snímané zařízením z kapitoly 2.2. Cílem je tedy nalézt pomocí trénovací sady kandidátní řešení, které zvládá korektně klasifikovat data z testovací sady. Každý dodaný soubor je zároveň ohodnocen experty z oblasti neurologie pro porovnání.

3.1 Hledání kandidátního řešení

Pro nalezení vhodného kandidátního řešení je tedy využit algoritmus CGP, se kterým jsme se seznámili v kapitole 2.1.4. Budeme tedy využívat schématu z obrázku 3.1, kde jednotlivé kroky budou popsány níže.

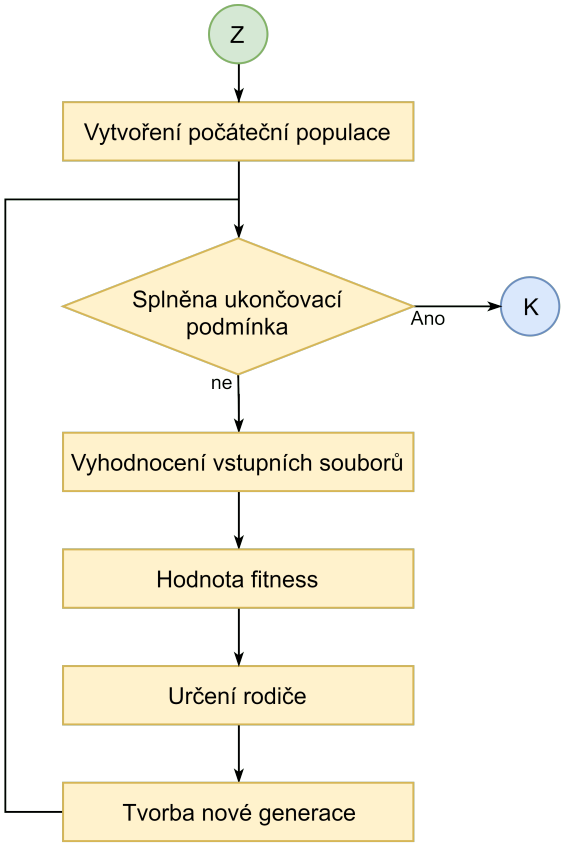
Pro experimenty pak bude zapotřebí možnost měnit parametry před spuštěním evoluce. Mezi tyto parametry patří parametry CGP z kapitoly 2.1.4, a to konkrétněji n_c, n_r a *L-back*, ale také velikost populace, množství mutací a ukončovací podmínka v podobě maximálního počtu generací.

3.1.1 Vytvoření počáteční populace

Pro vytvoření počáteční populace se nejdříve seznámíme s jednotlivými jedinci populace.

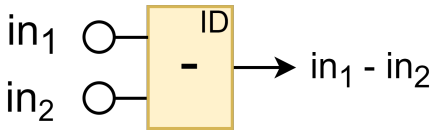
Jedinec Jedince si můžeme představit jako 2D tabulku $n_r \times n_c$, kde každý sektor nazýváme uzel. Uzlem je pak myšlena konkrétní funkce se dvěma vstupy. Takový uzel může vypadat jako na obrázku 3.2, kde ID určuje index daného uzlu, in_1 a in_2 jsou indexy argumentů funkce a znak – označuje operaci odčítání. Hodnota výstupu tohoto uzlu je tedy $in_1 - in_2$.

Každý uzel je tedy tvořen třemi čísly, které můžeme reprezentovat jako trojici, kde první a druhé označuje argumenty a třetí označuje index operace. Pro uzel na obrázku 3.2 můžeme jeho reprezentaci napsat například takto: $5, 7, 1$. Tímto říkáme, že prvním argumentem je uzel s indexem 5, druhým je uzel s indexem 7 a funkce má index 1 (v našem případě operace odčítání).



Obrázek 3.1: Popis algoritmu při návrhu klasifikátoru pomocí CGP.

Všechny uzly tabulky se pak dají zapsat jako řada $3 \cdot n_r \cdot n_c$ čísel. Pro dokončení celého jedince je ještě potřeba přidat adresy pro veškeré primární výstupy. Jedná se tedy pouze o indexy uzlů, na které jsou primární výstupy napojeny. Celkový řetězec čísel pro zaznamenání jedince se skládá z $3 \cdot n_r \cdot n_c + n_o$ čísel. V této práci budeme mít vždy pouze 1 primární výstup, tudíž $n_o = 1$, takže jedince zaznamenáme pomocí $3 * n_r * n_c + 1$ čísel.



Obrázek 3.2: Vizualizace jednoho uzlu chromozomu.

Populace Populací je pak myšleno n jedinců, kde n je velikost populace z předaných parametrů evoluce. Pro začátek evoluce je třeba vytvořit populaci z ničeho, proto se jednotliví jedinci generují náhodně. Pro každého jedince je tedy náhodně, ale dle pravidel uvedených v kapitole 2.1.4, vygenerován tento řetězec znaků. Jakmile máme jednotlivé jedince připravené, máme tím vytvořenou počáteční populaci. Nazveme ji tedy generací 0 a postupujeme podle algoritmu dále.

3.1.2 Ukončovací podmínka

Dalším krokem v našem algoritmu je ověření, zda nebyla splněna ukončovací podmínka. Podmínkou je myšlen maximální počet generací, tudíž podle aktuálního čísla generace určujeme, zda již proces ukončit či nikoli.

3.1.3 Vyhodnocení vstupních souborů

Pro celou generaci se postupně vyhodnotí každý chromozom na sadě trénovacích dat. Tím jsou myšleny csv soubory ze studie č. 1, které jsou popsány v odstavci 2.2. Každý soubor je pročítán pomocí plovoucího okna popsaného v sekci 2.2. Tudíž jako primární vstupy dostáváme vždy 32 hodnot. Výsledné ohodnocení chromozomu hledáme rekurzivně od primárního výstupu. Ze všech $N - 31$ ohodnocení se vypočítá aritmetický průměr.

Chromozom tedy postupně přiřadí konkrétní hodnotu ke každému souboru csv. Jakmile jsou ohodnoceny všechny soubory, hodnoty se seřadí podle velikosti. Toto pořadí se použije pro výpočet hodnoty fitness. Tento proces je opakován pro všechny chromozomy v generaci.

3.1.4 Fitness funkce

Jako fitness funkce je použita AUC (Area Under The Curve). [2] [3] [6]

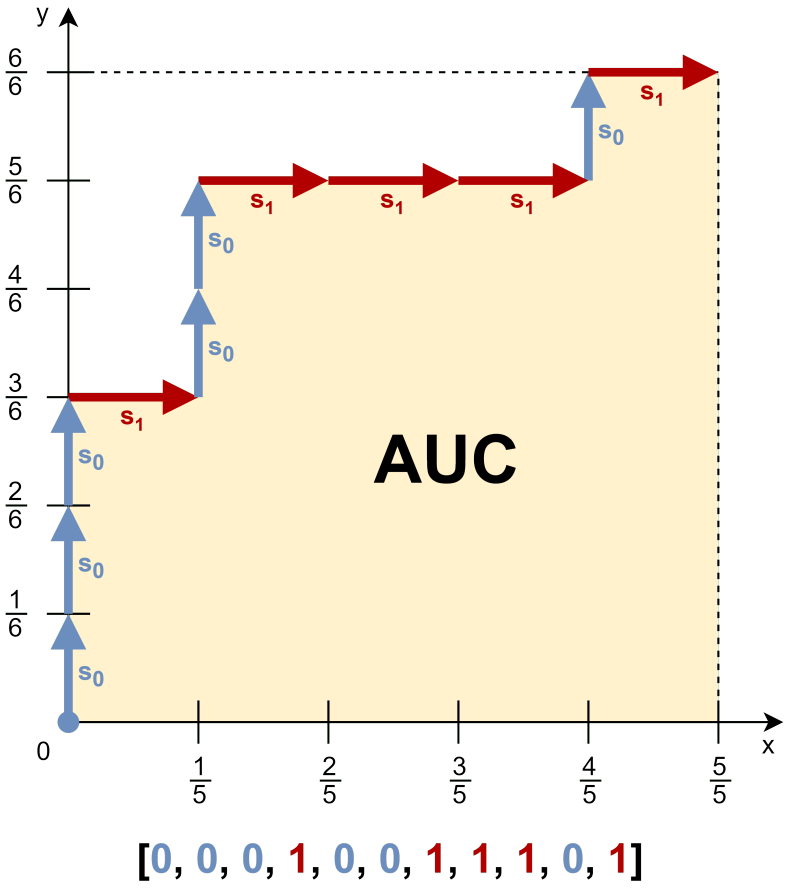
Hodnota AUC představuje schopnost binárního klasifikátoru rozlišit mezi třídami. Čím vyšší AUC, tím lepší model.

Hodnotou AUC je myšlena plocha pod ROC (Receiver Operating Characteristic - operační charakteristika přijímače) křivkou. Tato křivka se generuje pomocí řetězce složeného z n prvků, které mohou nabývat hodnot {0,1}. Kde n představuje počet zkoumaných dat (v našem případě počet trénovacích souborů). Každá hodnota prvku pak ukazuje, do které skupiny by měl daný prvek podle vstupních dat patřit. Perfektní klasifikátor dokáže perfektně oddělit jednotlivé prvky do dvou skupin.

Příklad - jako vstupní data máme $n = 7$ prvků, ze kterých by měly právě 4 padnout do skupiny '0' (*negativní*: $n_0 = 4$) a právě 3 do skupiny '1' (*pozitivní*: $n_1 = 3$). Klasifikátor po vyhodnocení jednotlivých prvků jejich ohodnocení seřadí podle velikosti. V tomhle pořadí je tedy očekáváno, že prvních n_0 prvků je negativních a zbylých n_1 prvků pozitivních. V perfektním případě by řetězec tohoto klasifikátoru vypadal takhle [0, 0, 0, 0, 1, 1, 1]. AUC nám popisuje, do jaké míry tomu ve skutečnosti je.

Pro vykreslení samotné křivky pak postupujeme následovně. Pokud tuto křivku chceme zakreslit do klasické kartézské soustavy souřadnic, bude naše křivka vždy vycházet z bodu [0,0] a končit v bodě [1,1]. $D(f) = H(f) =< 0, 1 >$. Definujeme si dvě hodnoty. Těmi jsou krok negativní $s_0 = \frac{1}{n_0}$ a krok pozitivní $s_1 = \frac{1}{n_1}$. Postupujeme podle řetězce z klasifikátoru. Pokud je prvek ohodnocen jako negativní, postoupí se o s_0 v kladném směru osy y. Pokud je prvek ohodnocen jako pozitivní, postoupí se o s_1 v kladném směru osy x. Na obrázku 3.3 můžeme vidět, jak se zakreslí křivka klasifikátoru, který vygeneroval řetězec [0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1]. Výsledná AUC je pak plocha pod touto křivkou. V tomto případě se jedná o AUC = 0,8.

Obecně pokud AUC roste dostáváme lepší výsledek. Tudíž pomocí CGP hledáme jedince s AUC nejblíže 1. Ovšem pokud najdeme jedince jehož AUC bude 0, znamená to pro nás, že klasifikátor funguje přesně obráceně. Tudíž rozřadí prvky perfektně, ale do špatných skupin. Kdyby se jen otočilo pořadí u tohoto klasifikátoru, vzniklo by nám opět perfektní řešení s AUC = 1. Proto pokud vyjde AUC v intervalu $< 0, \frac{1}{2})$, tak AUC = 1 - AUC. Díky této úpravě tedy říkáme, že nejhorší klasifikátor je klasifikátor s AUC = 0,5 a nejlepším



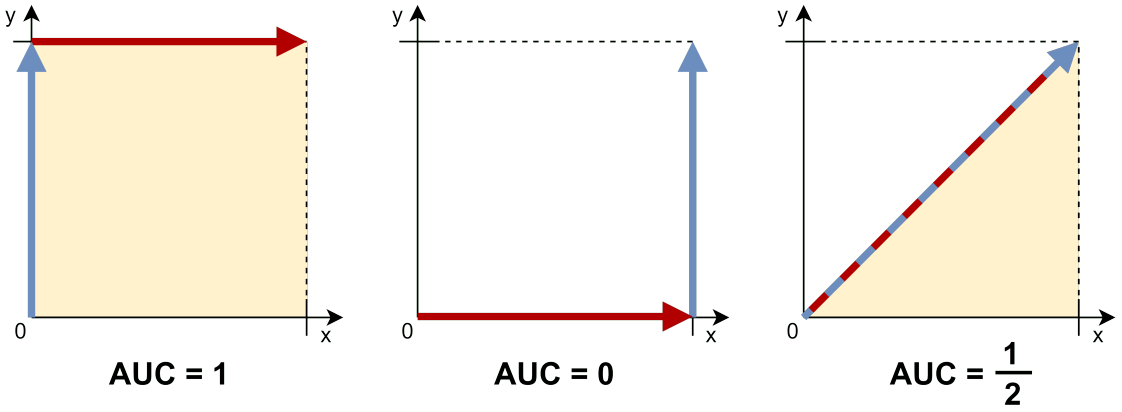
Obrázek 3.3: Příklad zobrazení AUC křivky. $s_0 = \frac{1}{6}$, $s_1 = \frac{1}{5}$, $AUC = 0,8$.

klasifikátorem je klasifikátor s AUC = 1 (resp. AUC = 0). Vizuálně tyto hodnoty AUC jsou zobrazeny na obrázku 3.4.

Pro trénování uvažujeme klasifikátor, který pouze rozlišuje mezi pacienty, kteří trpí dyskinezií a pacienty bez ní. Pro zefektivnění trénování je podle článku [4] vhodné brát jako vstupy pouze soubory ohodnocené úrovní 0 (pacienti bez dyskinezie) a soubory ohodnocené 3 a 4 (pacienti trpící závažnou dyskinezií). Díky tomu můžeme využít AUC, kde záznamy pacientů trpících dyskinezií zařadíme do skupiny 1 (pozitivní) a záznamy pacientů bez dyskinezie zařadíme do skupiny 0 (negativní).

3.1.5 Určení rodiče

Po ohodnocení všech jedinců generace můžeme porovnat jejich AUC a podle toho rozhodnout, který jedinec si v generaci vedl nejlépe. Pokud máme jednoznačného vítěze, stává se z něj rodič pro další generaci. Pokud máme více jedinců, co jsou ohodnoceni stejnou nejvyšší hodnotou AUC, vybírá se z nich ten, který nebyl rodičem v předchozí generaci. V této práci pracujeme vždy pouze s jedním rodičem.



Obrázek 3.4: AUC křivky s hodnotami $1, 0, \frac{1}{2}$.

3.1.6 Tvorba nové generace

Vybraný rodič je automaticky předán do následující generace. Zbytek populace pak tvoří potomci tohoto rodiče. Potomek vzniká mutací svého rodiče. Tudíž pokud chceme vytvořit nového potomka, zkopírujeme chromozom rodiče a zmutujeme do míry podle zadaného parametru. Tento parametr nám určuje, jak velká část chromozomu je potřeba zmutovat. Existují dvě varianty mutace. Buď máme pomocí parametru přesně určený počet mutovaných genů chromozomu a náhodně jen určujeme, o které se jedná. Nebo si pro každou hodnotu z chromozomu podle parametru určíme, zda má mutovat, či nikoli. V obou případech je třeba zachovat všechny podmínky jako při inicializaci, která je popsána v kapitole 2.1.4. Tímto máme vytvořenou novou generaci, tudíž navýšíme číslo generace a pokračujeme dále opět ukončovací podmínkou.

3.1.7 Ukončení evoluce

Ve chvíli, kdy je splněna ukončovací podmínka, se nejlepším chromozomem této evoluce stává rodič z poslední generace. Tento chromozom je teď potřeba otestovat na druhé sadě vstupních souborů, lépe řečeno data ze studie č. 2. Zde chceme zjistit, jak si tento chromozom vede při klasifikaci jednotlivých úrovní dyskinezie. Výsledkem jsou tedy 4 hodnoty AUC, kde byly postupně porovnány soubory jednotlivých úrovní dyskinezie se soubory s úrovní 0.

Kapitola 4

Implementace

Program implementuje standardní CGP algoritmus, napsaný v jazyce C. Program je postaven na funkcích, je tedy ryze funkcionální. Jedná se o konzolovou aplikaci, která pracuje se vstupními soubory a výstup vypisuje jak na standardní výstup tak do souboru. Jako parametry pro CGP jsou použity globální proměnné, které se zadávají při spouštění experimentů.

4.1 Parametry CGP

Při spuštění programu je třeba vhodně zvolit parametry. Pokud je deaktivována možnost interakce z uživatelem, spustí se program s defaultními hodnotami.

- **nRow** – Globální proměnná nRow představuje jeden rozměr v 2D tabulce uzlů chromozomu. Jedná se o integer. Jeho defaultní hodnota je 6. Uživatel je vyzván zadat tento parametr větou: "Pocet radku uzlu:".
- **nCol** – Globální proměnná nCol představuje druhý rozměr v 2D tabulce uzlů chromozomu. Jedná se o integer. Jeho defaultní hodnota je 6. Uživatel je vyzván zadat tento parametr větou: "Pocet sloupcu uzlu:".
- **lBack** – Globální proměnná lBack představuje míru propojitelnosti uzlů chromozomu. Jedná se o integer, kde jeho hodnota patří do intervalu $< 1, nCol >$. Jeho defaultní hodnota je 2. Uživatel je vyzván zadat tento parametr větou: "Mira propojitelnosti (L-back):".
- **mutationChance** – Globální proměnná mutationChance představuje procentuální pravděpodobnost, že se při mutaci konkrétní uzel změní. Jedná se o desetinné číslo (double) z intervalu $(0, 1 >$. Jeho defaultní hodnota je 0,08. Uživatel je vyzván zadat tento parametr větou: "Mira mutace (des. cislem):".
- **gyroskop** – Globální proměnná gyroskop ukazuje, zda se využijí pouze data z akcelerometru nebo data z gyroskopu. Jedná se o bool proměnnou, kde 0 (false) zastupuje akcelerometr a 1 (true) zastupuje gyroskop. Jeho defaultní hodnota je 0. Uživatel je vyzván zadat tento parametr větou: "Data z akcelerometru (0), data z gyroskopu (1):".
- **genSize** – Globální proměnná genSize představuje velikost populace v jedné generaci včetně rodiče. Jedná se o integer. Jeho defaultní hodnota je 5. Uživatel je vyzván zadat tento parametr větou: "Pocet jedincu v generaci:".

- **gensInRun** – Globální proměnná gensInRun představuje počet generací pro jednotlivé běhy programu. Jedná se o integer. Jeho defaultní hodnota je 500. Uživatel je vyzván zadat tento parametr větou: "Pocet generaci pro jeden beh:".
- **runNum** – Globální proměnná runNum představuje počet běhů, které program spustí. Jednotlivé běhy jsou spuštěny se stejnými parametry. Jedná se o integer. Jeho defaultní hodnota je 30. Uživatel je vyzván zadat tento parametr větou: "Pocet behu:".

4.2 Vstupní data

Jako vstupní data jsou použita data ze studií popsaných výše v části 2.2. Tuto soubory se vyskytují v adresářích se jmény “LID1data“ a “LID2data“. Jsou umístěny ve stejném adresáři jako samotný program. Jako další krok je tedy načíst tato data.

Načtení Každý z adresářů je pročítán zvlášť. V jednotlivých souborech představuje každý řádek jeden časový záznam pořízený pomocí snímajícího zařízení popsaného v kapitole 2.2. Každý časový záznam je složen z šesti hodnot. Celý soubor je pak reprezentován dvojrozměrným polem integerů. Jednotlivé záznamy souborů jsou pak uspořádány do pole souborů pro trénovací sadu *trainFilesDataArr* a pro testovací sadu *testFilesDataArr* . Zde je také potřeba ověřit, zda má soubor alespoň minimální požadovaný počet časových záznamů. Jelikož má chromozom 32 primárních vstupů, veškeré soubory obsahující méně než 32 časových záznamů jsou proto ignorovány.

Při načítání dat ze souboru se zároveň zaznamená, o jakou úroveň dyskinezie se podle expertů jedná. Tento posudek by se měl být součástí názvu souboru ve tvaru "Dyskinesia-X". Tuto vlastnost zpracovává funkce *readInputsNames* . Pokud se narazí na soubor, který má nekorektní název nebo obsahuje "Dyskinesia-undefined", je tento soubor ignorován, jelikož nám nedává žádná užitečná data. Počty jednotlivých ohodnocení jsou také uchovány v globálních proměnných pro pozdější využití.

4.3 Počáteční populace

Pro vytvoření populace je třeba vytvořit jedince. Jedincem je myšleno pole struktur *Node* (uzel), které jsou složeny ze tří proměnných. Tyto proměnné jsou *inp1* a *inp2* obsahující adresy vstupů a *operation* obsahující index operace. Uzlem jsou jak prvky tabulky, tak primární výstup. Pro vytvoření validního chromozomu je využita funkce *initRandomNode,* která postupuje podle podmínek v odstavci 2.1.4. Tato funkce je volána *genSize* -krát.

4.4 Trénování

V této fázi se používá pouze první sada dat pro nalezení vhodného řešení. Jak bylo řečeno výše v sekci 3.1.4, jako trénovací data jsou použity pouze záznamy s ohodnocením 0 (negativní) a 3, 4 (pozitivní). Proto při ohodnocování jedince záznamy 1 a 2 přeskakujeme.

4.4.1 Ohodnocení jedince

Postupně se vyhodnotí všichni noví jedinci. Pomocí plovoucího okna popsaného v části 2.2 se vezme 32 primárních vstupů, kterými jsou magnitudy tříosého akcelerometru nebo magnitudy tříosého gyroskopu. Podle parametru *gyroskop* určíme, které z nich. Následně

se zavolá rekurzivní funkce *computeValue* nad primárním výstupem. Tato funkce rozezná, o jaký uzel se jedná a podle toho buď vrací hodnotu primárního vstupu, vykoná operaci uzlu nad jeho argumenty nebo jen vrací první argument, pokud se jedná o primární výstup.

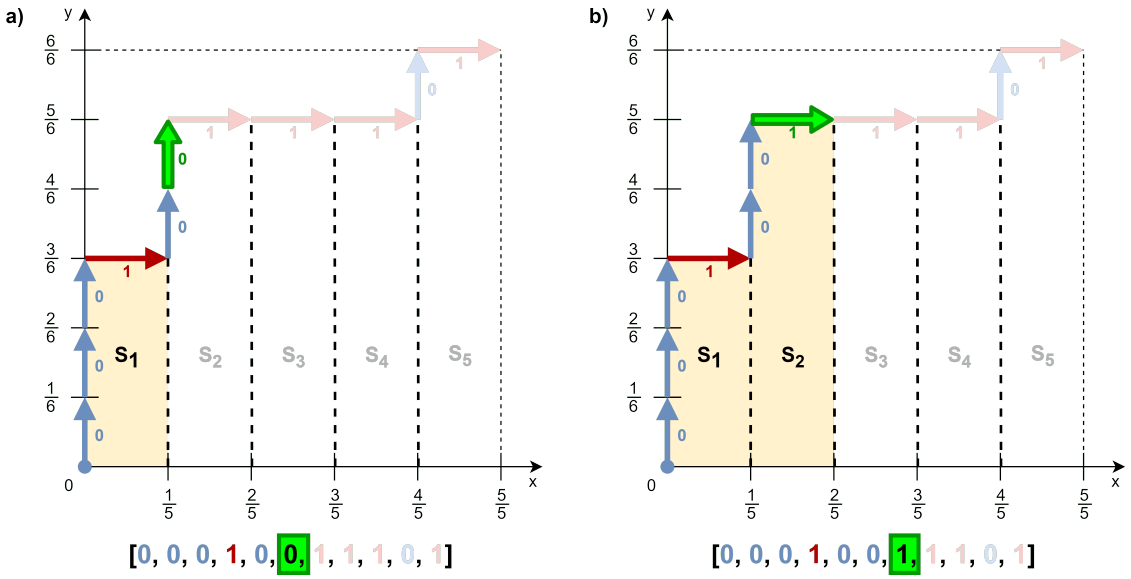
Po vyhodnocení pomocí plovoucího okna dostaneme pro jednotlivé soubory vždy $N-31$ ohodnocení. Tyto hodnoty se zprůměrují, čímž vzniká celkové ohodnocení jednoho souboru trénovacích dat. Jakmile jsou ohodnoceny všechny trénovací soubory, vypočte se hodnota fitness, neboli AUC.

AUC Pro výpočet AUC potřebujeme seřadit ohodnocení jednotlivých souborů podle velikosti. Pořadí indexů souborů je uloženo v poli *trainOrder*. Nyní chceme vypočítat plochu pod křivkou podle algoritmu v odstavci 3.1.4. O to se stará funkce *auc*. Této funkci je předán řetězec *trainOrder*, počet souborů ohodnocených 0 (negativní) a počet souborů ohodnocených 1 (pozitivní).

Vizuálně tato funkce rozdělí graf AUC křivky na obdélníky, jejichž obsahy počítá postupně. Stejně, jako na obrázku 3.3 určíme negativní a pozitivní krok, $s_0 = \frac{1}{n_0}$ a $s_1 = \frac{1}{n_1}$. Dále definujeme dvě proměnné $hight = 0$ a $S = 0$, které popisují funkční hodnotu aktuálního bodu a celkový obsah pod křivkou.

Jak je znázorněno na obrázku 4.1, pokud se v řetězci objeví 0, posouváme se o s_0 v kladném směru osy y. Navýšíme tedy hodnotu $hight \mathrel{+}= s_0$. Pokud se v řetězci objeví 1, posouváme se o s_1 v kladném směru osy x. Vzniká nám tak obdélník o výšce *hight* a šířce s_1. Obsah se vypočítá jako $S \mathrel{+}= hight \cdot s_1$.

Jako poslední krok ověříme, zda jsme nenašli AUC $< 0{,}5$ jak bylo popsáno v sekci 3.1.4. Pokud tomu tak je, funkce *auc* vrací hodnotu $1 - S$, jinak pouze S.



Obrázek 4.1: Znázornění výpočtu AUC.

4.4.2 Nová generace

Po ohodnocení všech jedinců je třeba vybrat rodiče pro další generaci. Porovná se tedy výsledná AUC u celé populace a vybere se nejlepší chromozom. Pokud je více jedinců

ohodnocených stejnou AUC, vybere se jedinec, který není označen jako rodič minulé generace (při první generaci není označen žádný). Tento vybraný chromozom se stává rodičem následující generace.

Vytvoření potomků Zvolený rodič se nemění, proto není třeba ho znovu ohodnocovat v další generaci. Potomci jsou generováni pomocí mutací, jak je popsáno v části 2.1.4. Pro všechny jedince, kteří nepřežili poslední generaci je generováno $3 \cdot nRow \cdot nCol + 1$ náhodných čísel, které určí, zda je daný gen v chromozomu zkopírován od rodiče nebo vygenerován nový. Opět je třeba dodržet veškerá pravidla napojení. Tímto je ukončen jeden generační cyklus. S novou generací se pokračuje stejným způsobem.

4.5 Testování

Jakmile je dosaženo *gensInRun* generací, nejlépe ohodnocený chromozom je pak řešením celého běhu. Tento chromozom je potřeba otestovat. K tomu se využije dat ze studie č. 2. Chromozom je testován celkem čtyřikrát, jelikož testujeme jeho schopnost identifikovat konkrétní úrovně dyskinezie. Proto jsou postupně poskytnuta data pouze ze souborů ohodnocených 0 a 1, tudíž testujeme, jak zvládá chromozom rozlišit pacienta bez dyskinezie oproti pacientovi s nejnižší úrovní dyskinezie. Stejně jako u trénování chromozomu je využita fitness funkce AUC popsaná v části 4.4.1, kde jako n_0 (negativní) bereme počet testovacích souborů označených "Dyskinesia-0" a n_0 (pozitivní) bereme počet testovacích souborů označených "Dyskinesia-1". Stejným způsobem je pak vypočítána hodnota AUC pro porovnání 0 a 2, 0 a 3, 0 a 4. Výsledkem běhu jsou tedy tyto 4 hodnoty AUC.

4.6 Ukončení běhu

Jakmile doběhne trénování populace i testování nejlepšího chromozomu porovná se číslo běhu s *runNum* . Pokud je *runNum* = 1, je program ukončen. V opačném případě je celý algoritmus opakován se stejnými parametry.

4.7 Výstup

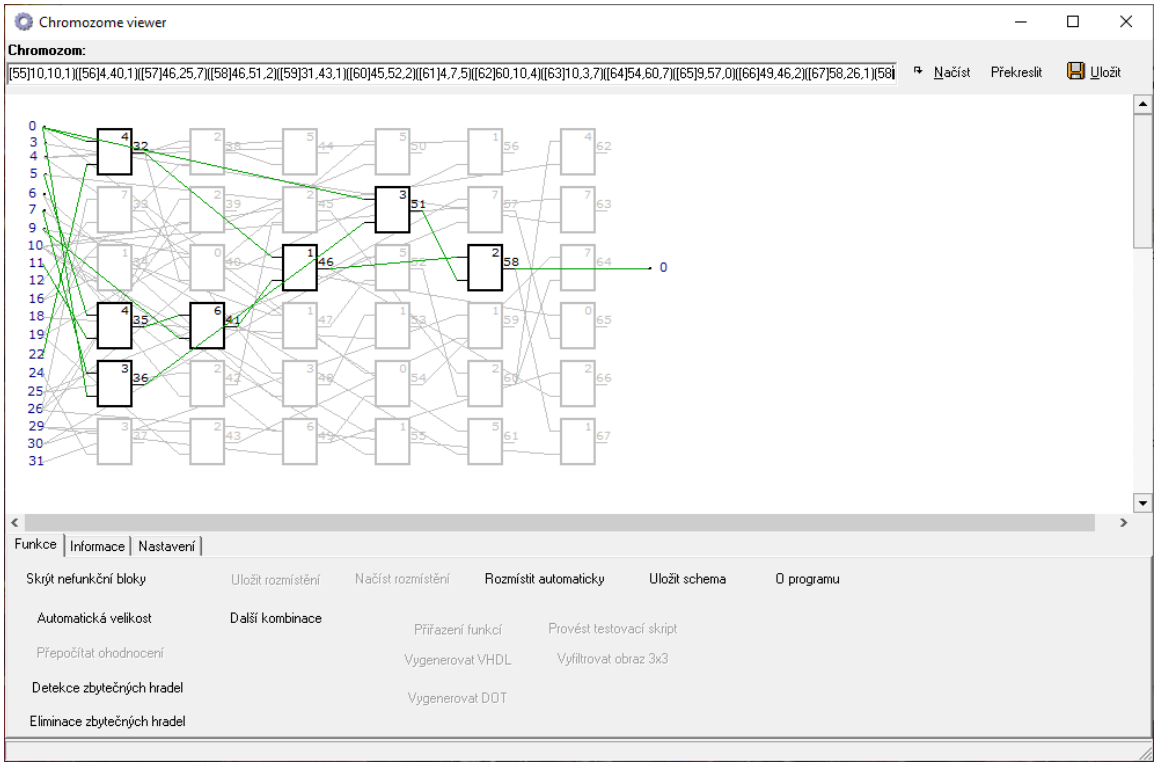
Veškerá užitečná data jsou zapisována do csv souboru *out.csv* , kde jeden řádek zastupuje veškeré výstupy jednoho běhu. Tudíž zápis probíhá vždy po ukončení jednoho běhu.

Informace, které program ukládá jsou:

- **nRow, nCol, lBack, genSize, mutationChance, gensInRun** – Parametry běhu.
- **timeStart** – Reálný čas začátku běhu (vytvoření počáteční populace).
- **timeOfBest** – Reálný čas, kdy bylo nalezeno první řešení se stejnou AUC jako výsledný chromozom.
- **timeEnd** – Reálný čas, kdy byl běh ukončen.
- **trainAUC** – Výsledná AUC pro trénovací skupinu dat.
- **path(0)(X)** – Celkem čtyři záznamy řetězce použitého k znázornění AUC testovací řady mezi 0-X.

- **testAUC(0)(X)** – Celkem čtyři hodnoty AUC pro porovnání 0-X
- **BestChrom** – Řetězec deklarující nejlepší chromozom běhu.

Způsob zápisu chromozomu Chromozom je vypisován způsobem, který je použitelný jako vstup do vizualizéru od docenta Zdeňka Vašíčka [9]. Tento nástroj je vhodný pro jednodušší vizualizaci CGP chromozomu.



Obrázek 4.2: Ukázka chromozomu pomocí CGP vieweru od doktora Zdeňka Vašíčka [9].

4.8 Konzolové výpisy

Jelikož je program konzolovou aplikací, využívá se i výpisu do konzole. Na obrázku 4.3 můžeme vidět příklad zadávání parametrů běhu po spuštění programu. Na obrázku 4.4 můžeme vidět průběžný výpis při dokončení každé generace.

4.9 Problémy a případná rozšíření

Největším problémem při implementaci byla nedostatečná informovanost o problematice. Proto je spousta problémů řešena neefektivně. Tato neefektivnost později způsobila dlouhé komputace generací. Například reprezentace chromozomu pouze pomocí jednoho pole integerů by ulehčilo spoustu práce a zjednodušilo náročnost s jeho manipulací. Tato změna by zároveň urychlila mutace, které zabírají nejvíce času díky jejich četnosti v programu. Zároveň bych změnil způsob mutace. Jak je uvedeno v části 3.1.6, pokud je předem určený počet genů, které se mají mutovat, stačí nám vygenerovat tento počet náhodných čísel, které určí

```
Pocet radku uzlu: 6
Pocet sloupcu uzlu: 6
Mira propojitelnosti (L-back): 0.08
Mira mutace (des. cislem): Data z akcelerometru (0), data z gyroskopu (1): 0
Pocet jedincu v generaci: 5
Pocet generaci pro jeden beh: 500
Pocet behu: 30

Parametry: nRow=6, nCol=6, lBack=0, mutationChance=0.080000, gyroskop=0, gensInRun=500, runNum=30


Loading from file ../LID1data:

Reading LID1 0%
Reading LID1 1%
Reading LID1 3%
Reading LID1 4%
Reading LID1 7%
Reading LID1 9%
Reading LID1 10%
Reading LID1 11%
```

Obrázek 4.3: Příklad zadávání parametrů běhu.

```
33: [25] + [24] 39: [28] / [22] 45: [11] - [4]  51: MIN[24][15] 57: [8] + [2]   63: [11] * [31]
34: [15] + [29] 40: MIN[31][18] 46: [3] * [1]   52: MIN[28][6]  58: MIN[7][3]   64: MAX[31][15]
35: MIN[7][13]  41: [23] / [23] 47: ABS[29][28] 53: [14] / [13] 59: [8] - [24]  65: MAX[26][27]
36: [17] + [9]  42: ABS[6][20]  48: MAX[2][12]  54: ABS[16][1]  60: AVG[12][25] 66: [7] * [24]
37: MIN[22][10] 43: MAX[17][31] 49: [31] + [28] 55: [6] / [5]   61: MAX[9][10]  67: ABS[14][13]
Output 68: [7]

-----------------------------------------------------------------------------------------------

AUC pro chromozom 0 v generaci 6 je 0.587169

AUC pro chromozom 1 v generaci 6 je 0.587169

AUC pro chromozom 2 v generaci 6 je 0.587169

AUC pro chromozom 3 v generaci 6 je 0.587169

AUC pro chromozom 4 v generaci 6 je 0.587169

Wed May  5 13:20:51 2021

Nejlepsi AUC za generaci 6 = 0.587169
Tabulka chromozomu 3:
32: [11] - [9]  38: AVG[22][22] 44: [1] - [13]  50: AVG[19][26] 56: MIN[4][14]  62: MAX[9][31]
33: [25] + [24] 39: [27] / [29] 45: [11] - [4]  51: MIN[24][15] 57: [8] + [2]   63: [11] * [31]
34: [15] + [29] 40: MIN[31][18] 46: [3] * [1]   52: MIN[28][6]  58: MIN[7][3]   64: MAX[31][19]
35: MIN[7][13]  41: [23] / [23] 47: ABS[29][28] 53: [14] / [13] 59: [8] - [24]  65: MAX[26][27]
36: [17] + [9]  42: ABS[6][20]  48: MAX[2][12]  54: ABS[16][1]  60: AVG[12][25] 66: [7] * [24]
37: MIN[22][10] 43: MAX[17][31] 49: [31] + [28] 55: [6] / [5]   61: MAX[28][10] 67: ABS[14][13]
Output 68: [7]

-----------------------------------------------------------------------------------------------
```

Obrázek 4.4: Výpis AUC všech chromozomů generace s podrobnou reprezentací nejlepšího z nich.

o které geny se jedná. Způsob použitý v této práci generuje náhodné číslo pro každý gen v chromozomu.

Jedním z rozšíření, které by dle mého názoru napomohlo k nalezení lepšího kandidátního řešení je využití jak dat z akcelerometru, tak i hodnot z gyroskopu. Například zdvojnásobit počet primárních vstupů, kde budou magnitudy obou záznamů.

Pro zvýšení efektivnosti hledání bych chtěl zavést možnost paralelizace pro rychlejší prohledávání a také pro možnost spouštění několika běhů o jiných parametrech zároveň.

Pro zjednodušení práce s programem bych chtěl navrhnout jednoduché uživatelské rozhraní. Pro jednodušší zadávání parametrů, s vizualizací progresu populace, s přehledným histogramem a vizualizací výsledného chromozomu.

Kapitola 5

Experimentální vyhodnocení

V této kapitole se budeme bavit o způsobu hledání vhodného klasifikátoru pomocí programu z kapitoly 4. Pro nalezení nejvhodnějšího klasifikátoru je potřeba vhodně zvolit parametry evolučního algoritmu. Ty jsou určeny experimentálně a jsou popsány dále v této kapitole. Pro trénování a testování tohoto klasifikátoru jsou použita data ze studií představených v části 2.2.

5.1 Popis vstupních dat

Jak již bylo zmíněno v sekci 4.2, pro trénování klasifikátoru jsou použita data ze studie č. 1. Data poskytnutá touto studií jsou naměřené záznamy šesti pacientů, pomocí snímacího zařízení představeného v části 2.2. Měření probíhalo v šestihodinových úsecích. Pro testovací skupinu dat byla využita data ze studie č. 2, kde měření podstoupilo sedmnáct pacientů po dvouhodinových úsecích měření.

V obou případech byli pacienti zároveň monitorováni experty v oblasti neurologie, kteří jednotlivé záznamy ohodnotili pomocí dosavadních praktik. Na obrázku 5.1 můžeme vidět četnosti zaznamenaných úrovní dyskinezie.

UDysRS	Study 1	Study 2
0	2933	1747
1	1227	971
2	1688	562
3	681	183
4	64	361

Obrázek 5.1: Četnosti jednotlivých úrovní dyskinezie ze studií č. 1 a č. 2. Obrázek převzat ze článku [4].

Pro hledání vhodného klasifikátoru byla použita pouze data ohodnocena úrovněmi 3 a 4, představující jednu třídu a data s úrovní 0, představující druhou třídu. Dle článku [4] měl tento způsob výrazně lepší výsledky. Vyřazením záznamů úrovní 1 a 2 byl velice zrychlen proces nalezení stejně kvalitního klasifikátoru, jako při použití všech úrovní.

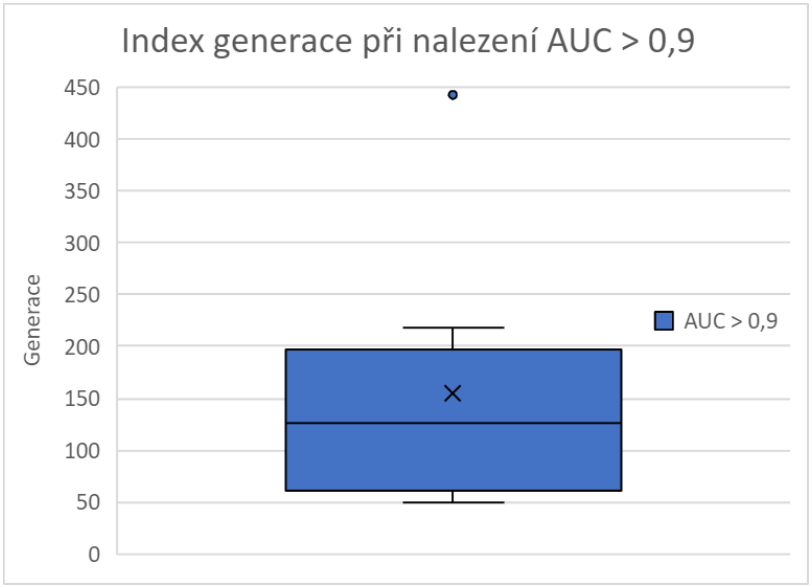
Jedná se o citlivá data pacientů, proto byly experimenty spouštěny vedoucí práce.

5.2 Hledání vhodného nastavení parametrů

Pro všechny parametry z pasáže 4.1 mají vliv na rychlost a efektivnost prohledávání stavového prostoru. Ve článku Michaela Lonese [4] bylo prohledávání spuštěno s parametry $nRow = 6$, $nCol = 6$, $genSize = 200$ a $gensInRun = 100$. Jelikož veškeré experimenty neprobíhají na superpočítači, velikost populace se bude muset výrazně snížit. Naopak můžeme navýšit počet generací pro nalezení vhodného řešení. První testovací běhy proběhly nad tabulkou 6×6, s parametry $lBack = 2$, $mutationChance = 0,05$, $gyroskop = 0$, $genSize = 7$ a $gensInRun = 1000$. Již zde bylo zřetelně vidět, že průběh jednoho běhu trvá příliš dlouho. Jelikož program nalezl *velmi dobré* řešení již po 200 generacích, bylo třeba tento parametr snížit. První experiment tedy provádíme za účelem odhadu vhodného počtu generací pro jeden běh. Dále se zjišťují vhodné hodnoty parametrů $genSize$, $lBack$ a $mutationChance$. Každý experiment byl pro jednotlivé hodnoty spuštěn minimálně desetkrát. Velikost tabulky byla ponechána stejná jako ve článku [4].

5.2.1 Experiment č. 1 - Počet generací v běhu

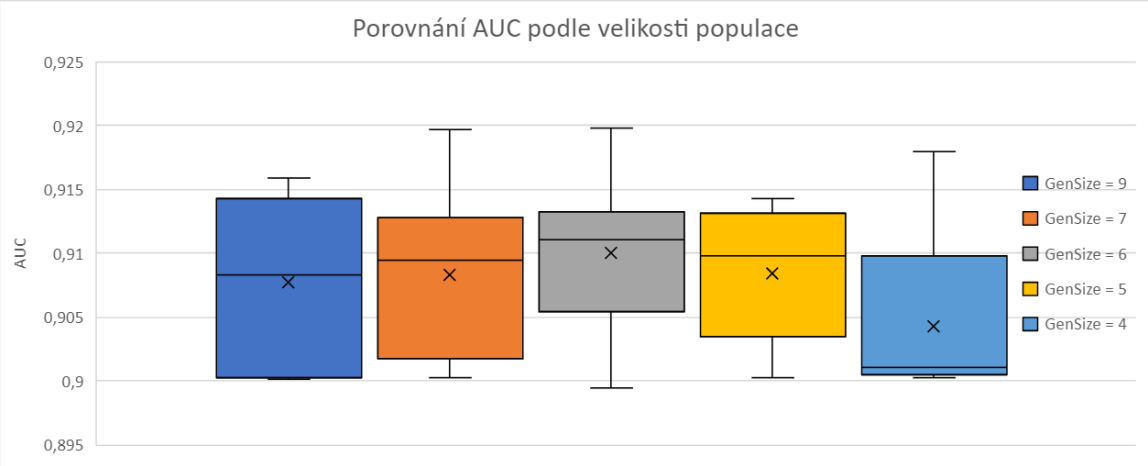
Pro nalezení vhodného parametru $gensInRun$ byl program spuštěn s parametry $lBack = 2$, $mutationChance = 0,08$, $gyroskop = 0$ a $genSize = 5$. Ukončovací podmínka pak byla upravena tak, aby přerušila běh ve chvíli, kdy je alespoň jeden chromozom ohodnocen AUC větší než daný práh a vypsala index generace, ve které bylo toto řešení nalezeno. Jako práh byla zvolena hodnota AUC = 0,9. Jak můžeme vidět na obrázku 5.2, požadované řešení bylo nalezeno v průměru již po 150. generaci. Pro nalezení i lepšího řešení, byla defaultní hodnota počtu generací navýšena na 500.



Obrázek 5.2: Krabicový graf znázorňující počet generací vhodných pro nalezení klasifikátoru ohodnoceného $AUC > 0,9$.

5.2.2 Experiment č. 2 - Velikost populace

Velikost populace také značně ovlivní rychlost prohledávání. Podle knihy [8] je vhodná velikost populace $genSize = 5$. Program byl proto spuštěn s parametry $lBack = 2$, $mutationChance = 0,08$, $gyroskop = 0$ a $gensInRun = 500$, kde jako ukončovací podmínka byla překročení konkrétního časového úseku. S dosavadními daty byl zvolen interval 4 hodin. Pro určení vhodné velikosti populace byly porovnány hodnoty AUC nejlépe ohodnocených chromozomů 5.3. Nejlepší průměrné ohodnocení měla jednoznačně hodnota $genSize = 6$.



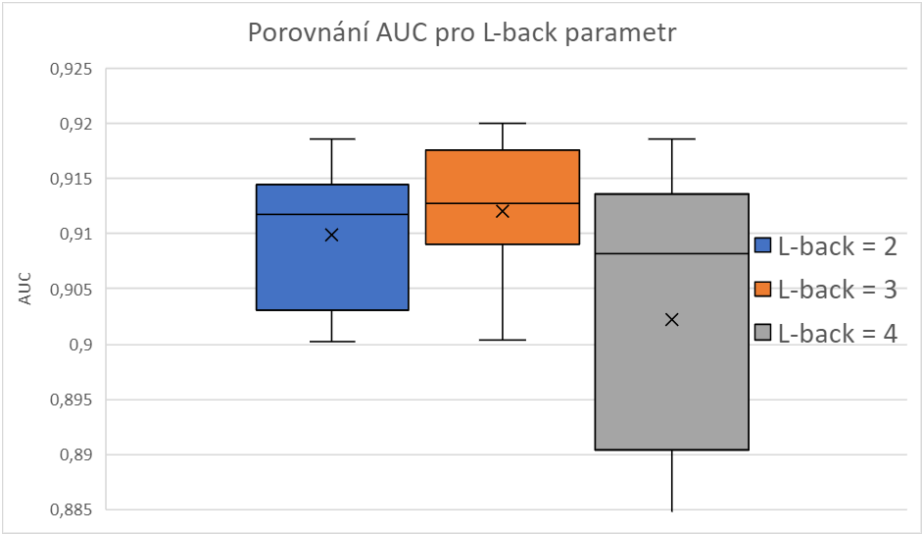
Obrázek 5.3: Krabicové graf znázorňující hodnotu AUC nejlepšího jedince pro různé hodnoty *genSize* .

5.2.3 Experiment č. 3 - Propojitelnost

Další experiment byl zaměřen na nalezení vhodného parametru *lBack* . Pokud budeme počítat se stejnými rozměry tabulky jako Michael Lones (6×6), jako L-back můžeme uvažovat pouze hodnoty 1 až 5. $lBack = 1$ povolí napojení pouze na jeden předchozí sloupec tabulky. Jedná se o minimální propojitelnost. Prohledávací prostor je tedy omezen na příliš složité genotypy. $lBack = 5$ je pak maximální propojitelnost, kde uzly mohou být napojeny na jakékoli uzly z předchozích sloupců. Prohledávací prostor je příliš velký pro efektivní nalezení vhodného řešení. Pro tento experiment byl program spouštěn s parametry $mutationChance = 0,08$, $gyroskop = 0$, $genSize = 6$ a $gensInRun = 500$. Porovnáváme tedy nejlepší ohodnocení jedince po 500 generací. Na obrázku 5.4 můžeme vidět nejlepší výsledky u $lBack = 3$, proto nastavujeme tuto hodnotu jako defaultní.

5.2.4 Experiment č. 4 - Mutace

Počet možných mutací při vytváření nové populace nám určuje různorodost populace, neboli míru odlišitelnosti oproti rodiči. S rostoucím počtem mutací roste prohledávaný prostor kolem aktuální maximální hodnoty. Vysoká mutace může zapříčinit rychlejší únik z lokálního maxima, ovšem opět rozšiřuje prohledávací prostor. Experiment tedy spustíme s parametry $lBack = 3$, $gyroskop = 0$, $genSize = 6$ a $gensInRun = 500$, kde porovnáme AUC ohodnocení nejlepších jedinců po 500 generací. S daty, které byly nasbírány, byla defaultní



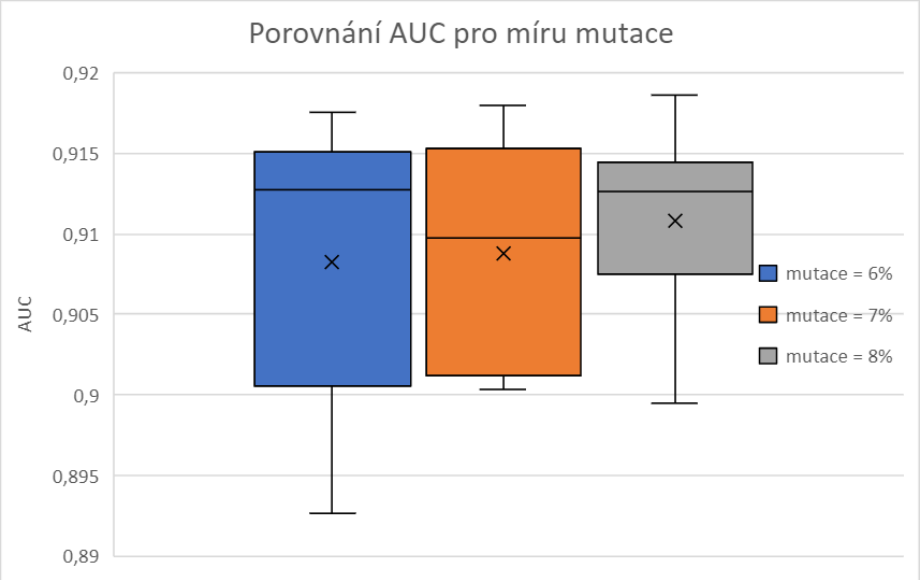
Obrázek 5.4: Krabicové grafy znázorňující hodnoty AUC nejlepšího jedince pro různé hodnoty *lBack* .

hodnota mutace nastavena na 8%. Výsledky experimentu jsou znázorněny v obrázku 5.5. Soupis všech vybraných parametrů je vidět na obrázku 5.6.

5.3 Hledání vhodného klasifikátoru

Pro nalezení vhodného klasifikátoru spustíme program s defaultními hodnotami $nRow = 6$, $nCol = 6$, $lBack = 3$, $mutationChance = 0,08$, $gyroskop = 0$, $genSize = 6$ a $gensInRun = 500$. Hned po ukončení prvního běhu byl nalezen klasifikátor *velmi dobré* kvality, $AUC = 0,920019$. Nejlepší řešení pro trénovací řadu bylo nalezeno při 25. běhu s ohodnocením $AUC = 0,921690$.

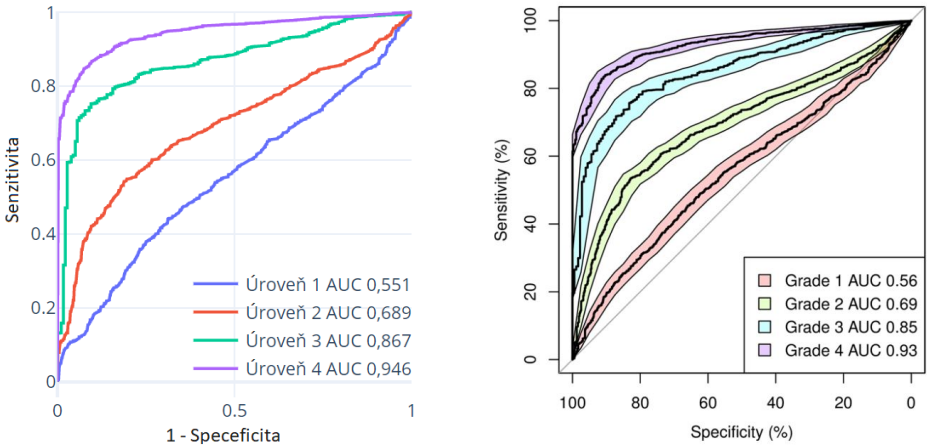
Při ukončení běhů byl vždy klasifikátor otestován i testovací sadou. Jako primární vstupy do nalezeného klasifikátoru jsou opět pomocí plovoucího okna brány magnitudy akcelerometru z dat studie č. 2. Toto testování je spuštěno celkově čtyřikrát, kde se klasifikátor snaží odlišit jednotlivé úrovně dyskinezie od záznamů pacientů bez dyskinezie. Pro první testování jsou tedy použity pouze záznamy s úrovněmi 0 a 1, u kterých se vyhodnotí AUC hodnota. Dále jsou jako primární vstupy použity hodnoty ze záznamů úrovní 0, 2, dále 0, 3 a nakonec 0, 4. Na obrázku 5.7 můžeme vidět jednotlivé AUC křivky pro všechny čtyři testy nejlépe ohodnoceného klasifikátoru na trénovací sadě v porovnání se stejným grafem klasifikátoru ze článku [4].



Obrázek 5.5: Krabicové grafy znázorňující hodnoty AUC nejlepšího jedince pro různé hodnoty *mutationChance* .

nRow x nCol	6 x 6				
gensInRun	500				
genSize	4	5	**6**	7	9
lBack	2		**3**		4
mutationChance	6%		7%		**8%**

Obrázek 5.6: Testované a experimentálně zvolené jako vhodné parametry pro spouštění programu.



Obrázek 5.7: Výsledky nalezeného nejlepšího klasifikátoru pro testovací sadu této práce (vlevo) a pro porovnání výsledky nejlepšího klasifikátoru ze článku [4] (vpravo).

Kapitola 6

Závěr

Hlavním cílem této práce bylo navrhnout a implementovat program za pomocí standardního kartézského genetického programování (CGP), který dokáže vhodně navrhnout klasifikátor úrovní dyskinezie. Navržený klasifikátor zvládá klasifikovat naměřená pohybová data pacientů do pěti skupin. Tento klasifikátor byl porovnán s klasifikátorem vygenerovaným pomocí implicitní kontextové reprezentace kartézského programování (IRCGP) ze článku Michaela Lonese [4] z pohledu schopnosti rozlišit mezi třídami (AUC). Ačkoli byla v této práci použita klasická varianta CGP, povedlo se najít klasifikátor, který dosahuje srovnatelných výsledků jako klasifikátor vygenerován pomocí IRCGP. Pro klasifikaci nejvyšší úrovně dyskinezie bylo dokonce nalezeno řešení lepší a to s AUC = 0,949212.

Díky této práci jsem nyní blíže seznámen s problematikou genetického programování, což mi otevřelo možnosti dalšího studia a budoucí profese. Tato problematika je pro mě velice zajímavá a rád bych v ní pokračoval i v budoucnu. Jelikož jsem psal technickou zprávu letos poprvé, dala mi tato práce důležité znalosti pro psaní prací v budoucnu.

Jelikož implementace CGP v této práci nepřebírá žádně cizí části kódu, některé pasáže by mohly být zjednodušeny. Pří pokračování v této práci bych rád zrychlil dobu komputace jedné generace a navýšil tak rychlost prohledávání. Dále bych rád využil i dat z gyroskopu. Tato implementace sice podporuje možnost nahrazení dat akcelerometru gyroskopem, ale nedovoluje využití obou druhů dat. Díky časovému nátlaku jsem nebyl schopen při této práci využít paralelizace, která by několikanásobně zrychlila proces vyhledávání. Rád bych tuto možnost přidal při dalším rozšiřování práce společně s jednoduchým uživatelským rozhraním.

Pokud by se mi podařilo dostatečně zefektivnit algoritmus, dal by se přestavět na obecný algoritmus CGP, který by se dal použít i v jiných oblastech.

Literatura

[1] Darwin, C. *On the Origin of Species by Means of Natural Selection Or the Preservation of Favoured Races in the Struggle for Life.* International Book Company, 1859.

[2] Hale, J. The 3 Most Important Basic Classification Metrics. Květen 2020. Dostupné z: https://towardsdatascience.com/the-3-most-important-basic-classification-metrics-3368dd425f74.

[3] Koehrsen, W. Beyond Accuracy: Precision and Recall. Březen 2018. Dostupné z: https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c.

[4] Lones, M. A., Alty, J. E., Cosgrove, J., Duggan Carter, P., Jamieson, S. et al. A New Evolutionary Algorithm-Based Home Monitoring Device for Parkinson's Dyskinesia. *Journal of Medical Systems.* 1. vyd. 2017, sv. 41, č. 11. ISSN 1573-689X.

[5] Miller, J. F. Cartesian Genetic Programming. In: *Cartesian Genetic Programming.* 1. vyd. Springer, 2011, kap. 2. ISBN 978-3-642-17309-7.

[6] Narkhede, S. Understanding AUC - ROC Curve. Červen 2018. Dostupné z: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[7] Popovici, E. Coevolutionary Principles. 2012. ISSN 987-1033.

[8] Sekanina, L. a kolektiv. *Evoluční hardware: Od automatického generování patentovatelných invencí k sebemodifikujícím se strojům.* 1. vyd. Academia, 2009. ISBN 978-80-200-1729-1.

[9] Vašíček, Z. *Tools4CGP* [online], 29. října 2008. [Cit. 8.5.2021]. Dostupné z: http://www.fit.vutbr.cz/~vasicek/cgp/tools/.

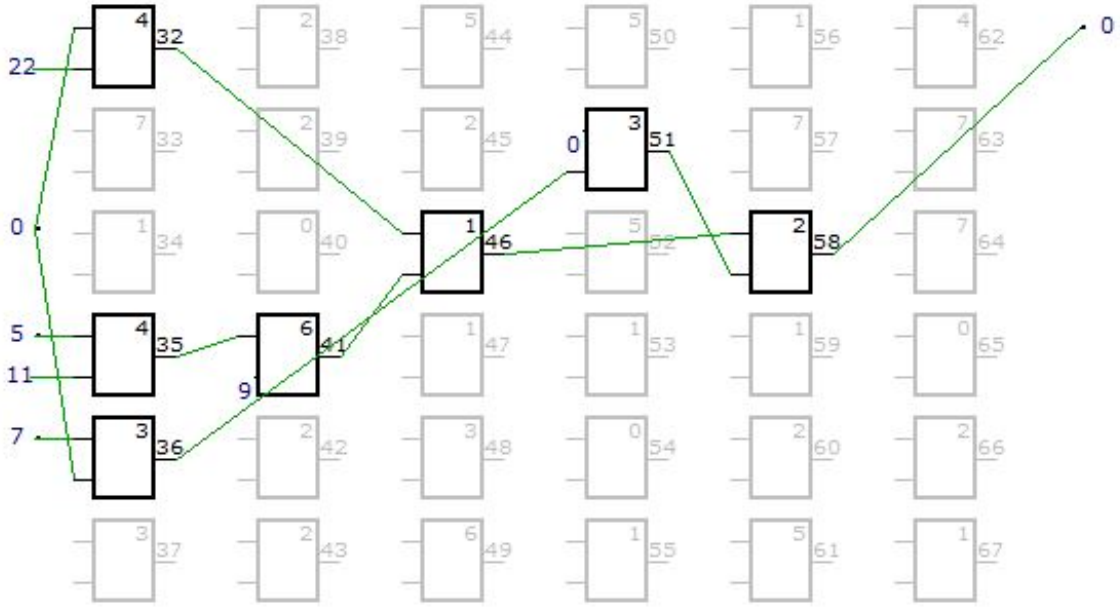
Příloha A

Nejlépe ohodnocené chromozomy

Trénovací sada

Nejlépe ohodnocený chromozom s primárními vstupy z trénovací sady s **AUC = 0,92169.** Vykreslený chromozom na obrázku A.1.

Číselný zápis: 0,22,4; 26,12,7; 19,6,1; 5,11,4; 7,0,3; 30,24,3; 32,16,2; 29,26,2; 19,32,0; 35,9,6; 25,9,2; 29,19,2; 4,38,5; 5,42,2; 32,41,1; 34,41,1; 33,6,3; 10,29,6; 26,16,5; 0,36,3; 30,47,5; 48,18,1; 0,49,0; 10,10,1; 4,40,1; 46,25,7; 46,51,2; 31,43,1; 45,52,2; 4,7,5; 60,10,4; 10,3,7; 54,60,7; 9,57,0; 49,46,2; 58,26,1; 58



Obrázek A.1: Chromozom s nejvyšší hodnotou AUC = 0,92169 pro trénovací vstupní data.

Testovací sada

Nejlépe ohodnocený chromozom s primárními vstupy z testovací sady, konkrétně klasifikace dyskinezie úrovně 1, dosahuje **AUC = 0,571389** . Vykreslený chromozom na obrázku A.2.

Číselný zápis: 18,9,5; 22,21,2; 12,11,5; 18,10,2; 2,29,5; 16,8,5; 2,8,3; 12,13,2; 30,24,3; 22,3,4; 17,27,4; 29,2,3; 21,42,4; 37,40,1; 37,29,6; 2,9,1; 2,9,0; 31,8,7; 39,46,0; 49,16,2; 30,7,7; 13,30,3; 24,1,0; 6,17,6; 11,15,0; 55,44,5; 16,27,4; 12,53,5; 4,20,2; 55,44,0; 57,15,1; 61,18,0; 4,60,4; 59,60,7; 3,5,7; 22,60,2; 62

Nejlépe ohodnocený chromozom s primárními vstupy z testovací sady, konkrétně klasifikace dyskinezie úrovně 2, dosahuje **AUC = 0,710051** . Vykreslený chromozom na obrázku A.3.

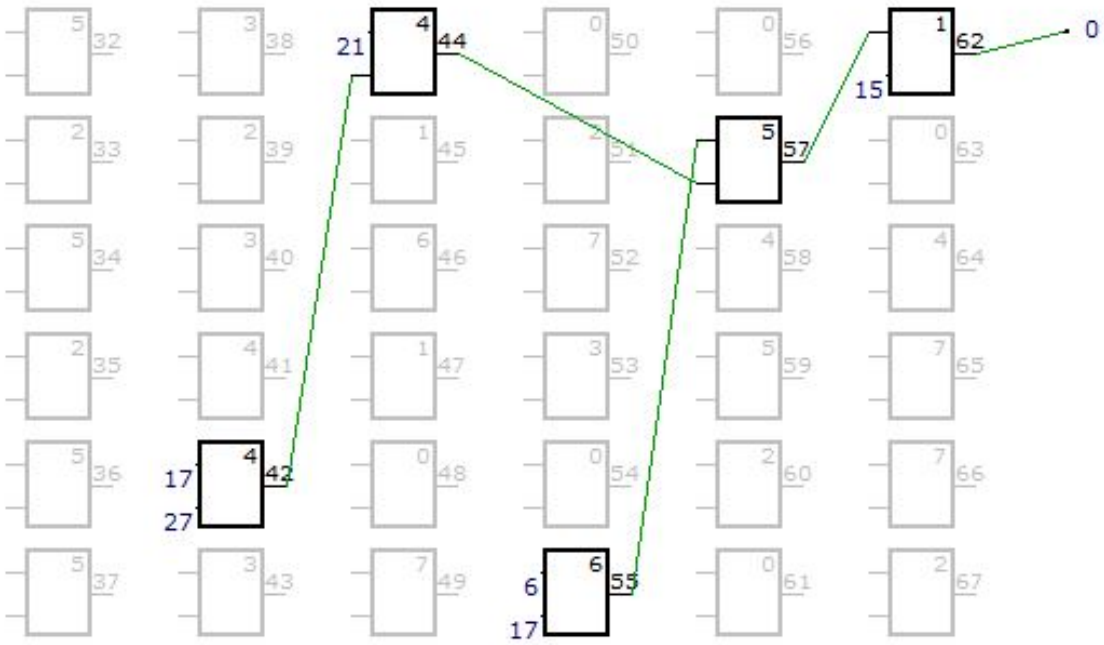
Číselný zápis: 0,6,6; 28,23,0; 12,30,7; 21,26,3; 18,18,5; 25,16,3; 23,7,1; 28,27,7; 19,18,1; 1,5,1; 19,10,0; 16,34,1; 30,42,0; 3,40,7; 37,3,6; 18,39,7; 27,19,6; 12,31,6; 24,21,4; 28,6,1; 14,47,2; 1,49,1; 25,1,2; 24,46,4; 48,5,2; 9,3,4; 2,26,7; 6,44,3; 51,8,4; 9,7,2; 60,59,7; 16,61,4; 10,29,2; 27,10,4; 12,26,3; 12,23,2; 62

Nejlépe ohodnocený chromozom s primárními vstupy z testovací sady, konkrétně klasifikace dyskinezie úrovně 3, dosahuje **AUC = 0,874692** . Vykreslený chromozom na obrázku A.4.

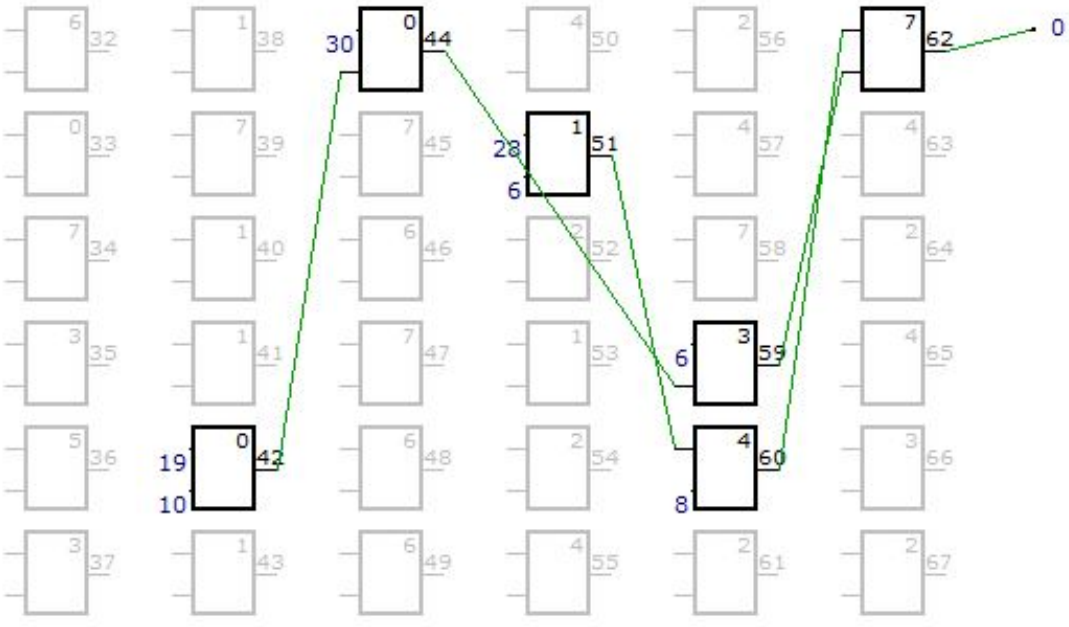
Číselný zápis: 8,30,1; 12,3,4; 22,25,5; 25,11,4; 3,30,4; 18,25,2; 34,8,0; 7,26,6; 0,34,7; 13,20,7; 32,33,2; 24,2,6; 12,4,4; 38,38,0; 16,3,4; 34,42,6; 26,18,3; 16,31,7; 13,47,5; 25,17,1; 31,29,2; 3,6,1; 38,41,2; 10,27,3; 44,15,6; 24,8,7; 23,5,4; 50,18,6; 45,53,1; 12,28,5; 26,50,3; 59,60,3; 61,25,7; 7,14,3; 0,20,1; 15,31,7; 63

Nejlépe ohodnocený chromozom s primárními vstupy z testovací sady, konkrétně klasifikace dyskinezie úrovně 1¨4, dosahuje **AUC = 0,949212** . Vykreslený chromozom na obrázku A.5.

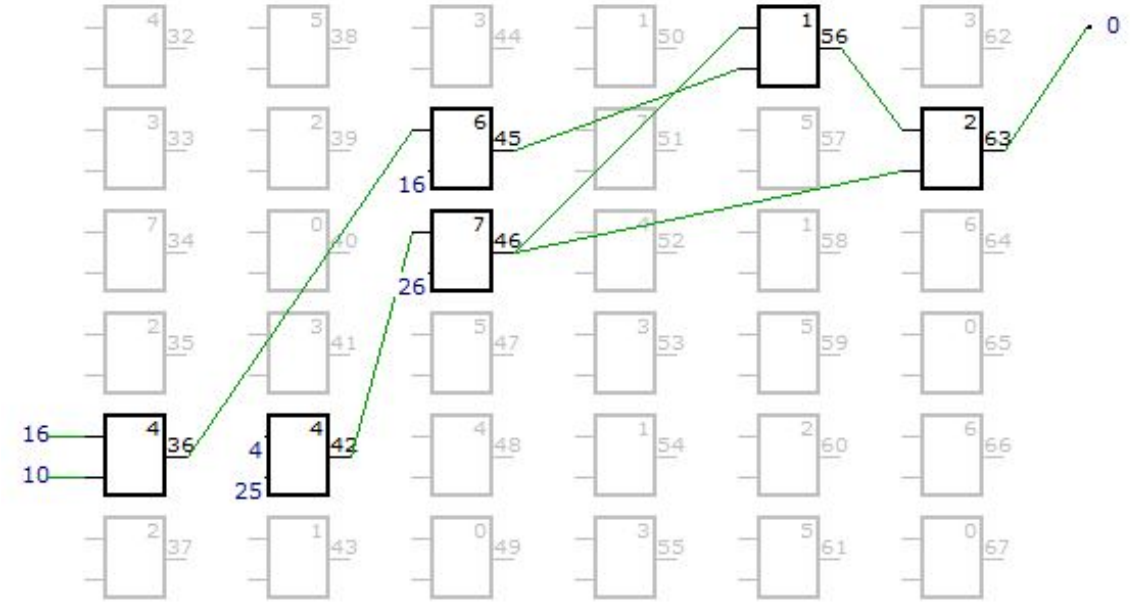
Číselný zápis: 17,13,5; 11,26,4; 28,3,0; 31,28,6; 8,14,5; 31,12,2; 14,18,6; 0,16,7; 8,11,1; 4,35,7; 35,36,4; 15,16,0; 8,15,5; 12,35,1; 26,42,1; 12,7,3; 14,41,1; 34,31,3; 38,33,3; 11,12,3; 38,21,2; 46,31,2; 6,47,6; 30,13,7; 20,20,6; 11,4,0; 50,1,0; 15,26,3; 0,49,4; 12,40,2; 50,10,7; 61,53,6; 54,16,1; 52,4,4; 8,47,6; 2,53,0; 63



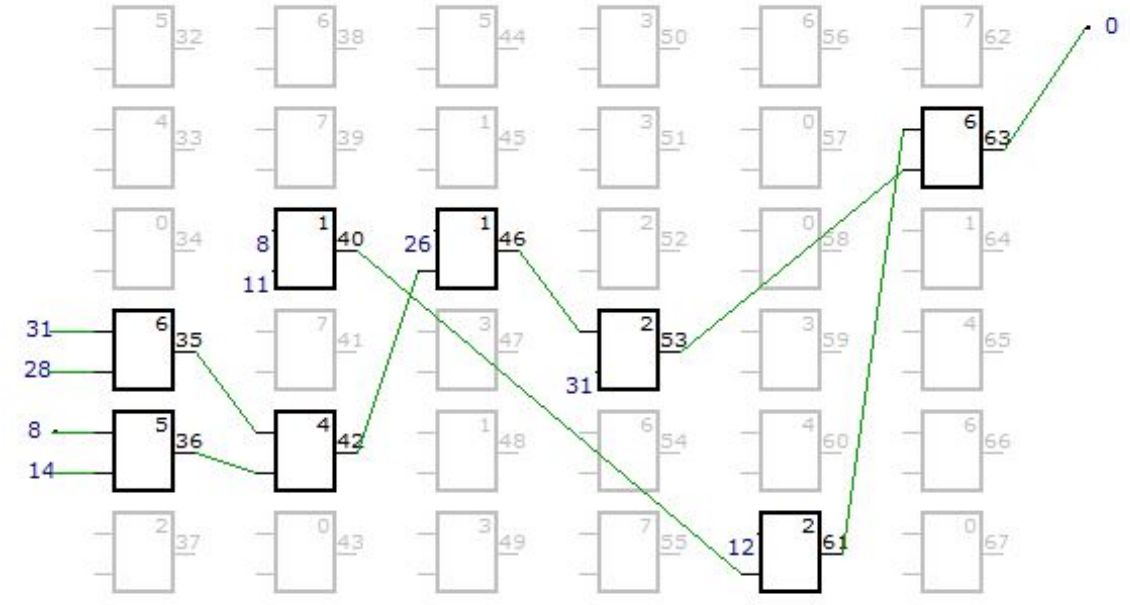
Obrázek A.2: Chromozom s nejvyšší hodnotou AUC = 0,571389 pro klasifikaci dyskinezie úrovně 1.



Obrázek A.3: Chromozom s nejvyšší hodnotou AUC = 0,710051 pro klasifikaci dyskinezie úrovně 2.



Obrázek A.4: Chromozom s nejvyšší hodnotou AUC = 0,874692 pro klasifikaci dyskinezie úrovně 3.



Obrázek A.5: Chromozom s nejvyšší hodnotou AUC = 0,949212 pro klasifikaci dyskinezie úrovně 4.