Excel@FIT 2015
http://excel.fit.vutbr.cz

Souběžné učení v koevolučních algoritmech

Michal Wiglasz*

Abstrakt
Kartézské genetické programování (CGP) se využívá zejména pro automatizovaný návrh číslicových obvodů, ale ukázalo se být úspěšnou metodou i pro řešení jiných inženýrských úloh. Časově nejnáročnější částí výpočtu je vyhodnocení kvality kandidátních řešení. Bylo ukázáno, že evoluci je možné urychlit pomocí koevoluce s prediktory fitness, které slouží k přibližnému určení kvality kandidátních řešení. Nevýhodou koevoluce je nutnost provést mnoho časově náročných experimentů pro určení nejvýhodnější velikosti prediktoru pro daný problém. V tomto článku je představena nová reprezentace prediktorů fitness s plastickým fenotypem, založená na principech souběžného učení v evolučních algoritmech. Plasticita fenotypu umožňuje odvodit různé fenotypy ze stejného genotypu. Díky tomu je možné adaptovat velikost prediktoru v průběhu evoluce na obtížnost řešeného problému. Z experimentů vyplývá, že lze dosáhnout srovnatelné kvality jako u standardního CGP při kratší době běhu programu a zároveň odpadá nutnost hledání nejvýhodnější velikosti prediktoru.

Klíčová slova: Koevoluční algoritmus — Kartézské genetické programování — Genetický algoritmus — Plasticita fenotypu — Predikce fitness — Obrazový filtr

Přiložené materiály: Zdrojové kódy

*xwigla00@stud.fit.vutbr.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

1. Úvod

Genetické programování umožňuje automatizovaně vytvářet programy s požadovaným chováním, které je definováno trénovací množinou. Každý prvek této množiny reprezentuje jeden případ fitness a skládá se ze vstupních hodnot programu a požadovaných výstupních hodnot [8]. U složitějších problémů může být takových případů fitness velmi mnoho. Protože výpočet fitness je časově nejnáročnější částí evoluce, je vhodné do trénovací množiny zahrnout pouze část případů fitness a urychlit tak výpočet. Na druhou stranu je třeba velikost množiny a používané případy fitness zvolit tak, aby byl program dostatečně obecný a i pro kombinace vstupů neobsažené v trénovací množině byly výstupy programu dle požadavků.

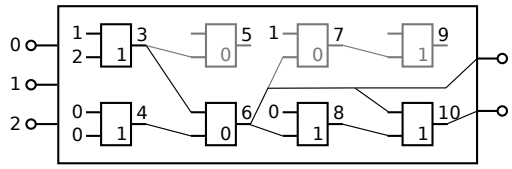
Vhodné podmnožiny případů fitness lze zvolit například pomocí genetického algoritmu. Jejich evoluce může probíhat souběžně s hledáním řešení problému – v minulosti byla představena koevoluce s prediktory fitness v genetickém progamování se stromovou reprezentací programů [3] i v kartézském genetickém programování (CGP) na úloze symbolické regrese [7]. Na úloze obrazových filtrů se ukázalo, že je potřeba použít prediktory s více případy fitness než u symbolické regrese, aby byla nalezená řešení dostatečně kvalitní [6]. Obecně platí, že před použitím algoritmu v nové úloze je třeba provést velké množství experimentů pro určení nejvýhodnější velikosti. Tento nedostatek lze řešit například nepřímým kódováním prediktorů, kdy jsou prediktory kódovány jako funkce generující posloupnost ukazatelů do trénovací množiny [5].

Cílem této práce je navrhnout a experimentálně vyhodnotit koevoluci CGP a přímo kódovaných prediktorů fitness s proměnlivou velikostí. Ta se adaptuje na složitost řešeného problému v průběhu evoluce kandidátních programů pomocí souběžného učení. Tímto se odstraní potřeba časově náročného experimentování s cílem nalézt co nejvýhodnější velikost prediktoru.

Tento článek má následující strukturu: v části 2 je představeno kartézské genetické programování a koevoluce s prediktory fitness. V části 3 je představeno nové plastické kódování prediktoru a způsob adaptace jejich velikosti na řešený problém. Část 4 se zabývá implementací navrženého algoritmu a jeho optimalizací pomocí vektorových instrukcí SSE2 a AVX2. V části 5 je implementovaný algoritmus experimentálně vyhodnocen a srovnán se standardním CGP a s koevolučním CGP s prediktory fixní velikosti.

2. Koevoluce prediktorů fitness v CGP

Kartézské genetické programování představil Julian Miller koncem devadesátých let. V CGP se kandidátní programy kódují jako orientované acyklické grafy, které jsou reprezentované dvourozměrnou kartézskou mřížkou výpočetních uzlů (funkčních bloků) o rozměrech $n_c \times n_r$. Příklad kartézského programu je na obrázku 1. Program má n_i primárních vstupů a n_o primárních výstupů. Každý ze vstupů funkčních bloků může být připojen k výstupu bloku o až l-back sloupců nalevo nebo na některý z primárních vstupů programu. Každý z výpočetních uzlů provádí jednu z n_a funkcí, množina těchto funkcí je volena s ohledem na řešený problém. Každý primární výstup programu je připojen na výstup některého z funkčních bloků [2].



Obrázek 1. Reprezentace programu v kartézském genetickém programování.

CGP bylo úspěšně použito v celé řadě úloh, jako je například symbolická regrese, klasifikace, návrh kombinačních obvodů nebo návrh obrazových filtrů. V případě obrazových filtrů je na vstup kartézského programu přivedeno zvolené okolí zpracovávaného pixelu a výstupem je filtrovaný pixel. Jako fitness funkce se často používá špičková hodnota poměru signál/šum v decibelech (Peak Signal to Noise Ratio, PSNR):

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{MN}\sum_{i,j}(v(i,j)-w(i,j))^2} \qquad (1)$$

kde M a N označují rozměry obrázku, v filtrovaný a w cílový (nepoškozený) obrázek [4].

Pro výpočet fitness kandidátního filtru musíme zpracovat celý obrázek, přičemž každý pixel lze považovat za jeden případ fitness. Protože i pro poměrně malé obrázky jich je několik tisíc, je výpočet fitness časově velmi náročný.

Tento problém se dá řešit pomocí koevolučního algoritmu. Při koevoluci se souběžně vyvíjí několik různých populací, které mezi sebou interagují prostřednictvím fitness funkcí. Fitness jedince závisí kromě jeho fenotypu také i na jedincích z okolních populací. V případě koevoluce s prediktory fitness existují dvě populace a archiv, do kterého jsou vkládána nejlepší nalezená kandidátní řešení (viz schéma na obrázku 2). Ta jsou vyvíjena pomocí CGP a jako fitness funkce slouží predikovaná fitness $f_{predicted}$, která je určená pomocí nejlepšího nalezeného prediktoru. Po vložení nového jedince do archivu se vypočítá i jeho skutečná fitness f_{exact}, pomocí které se ohodnocují prediktory.

Ve druhé populaci se pomocí genetického algoritmu vyvíjí prediktory fitness, což jsou podmnožiny množiny případů fitness (pixelů trénovacího obrázku). Jejich chromozomy jsou vektory ukazatelů do trénovací množiny o konstantní délce. Potomci jsou tvořeni pomocí křížení a mutace, kvůli zachování diverzity populace je několik potomků vytvářeno zcela náhodně. Využívá se i elitismus, kdy několik nejlepších jedinců rodičovské populace přechází do populace potomků. Fitness prediktoru je určena jako střední absolutní odchylka skutečné a predikované fitness všech programů v archivu:

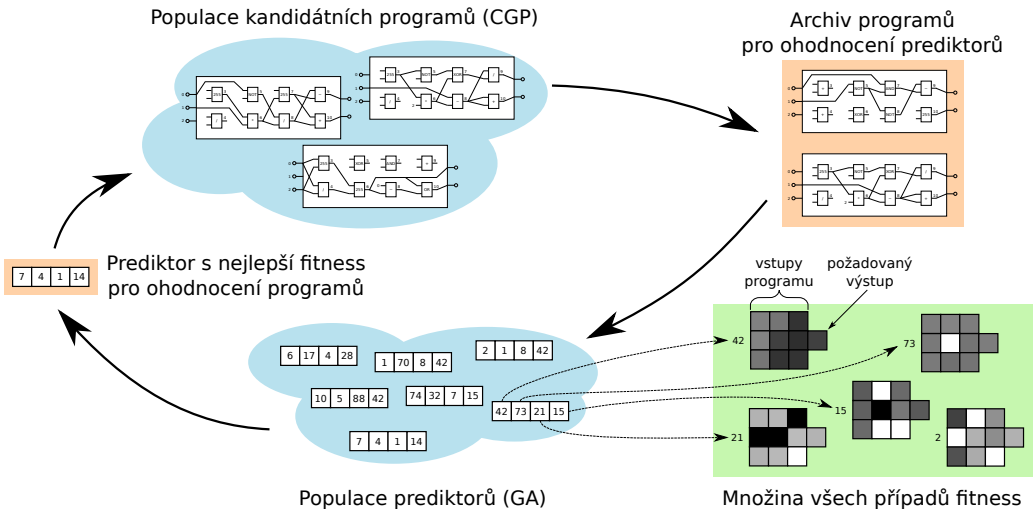
$$f(p) = \frac{1}{u}\sum_{i=1}^{u}\left|f_{exact}(s(i)) - f_{predicted}(s(i))\right| \qquad (2)$$

kde p označuje prediktor a s archiv programů, který obsahuje u jedinců. Prediktor s nejnižší odchylkou je pak použit pro ohodnocování kandidátních filtrů.

Po spuštění programu jsou náhodně vytvořeny počáteční populace filtrů a prediktorů fitness. Poté je vypočtena skutečná fitness filtrů a nejlepší jedinec je umístěn do archivu. Pomocí něj jsou následně ohodnoceny prediktory fitness a je určen nejlepší z nich, který bude zpočátku sloužit pro výpočet predikované fitness. Po této inicializaci archivů pokračuje běh programu ve dvou vláknech, ve kterých probíhá evoluce jednotlivých populací [6].

3. Adaptivní velikost prediktorů fitness

Nevýhodou koevolučního přístupu je nutnost předem stanovit velikost prediktoru, přičemž pro různé úlohy



Obrázek 2. Schéma koevoluce CGP a prediktorů fitness.

je výhodnější použít jinou velikost. Před použitím koevolučního CGP pro řešení nového problému je pak někdy nutné provést celou řadu časově náročných experimentů, jejichž cílem je určit nejvýhodnější velikost prediktoru. Cílem této práce je navrhnout, implementovat a experimentálně vyhodnotit algoritmus pro návrh obrazových filtrů, který využívá koevoluci s přímo kódovanými prediktory, jejichž velikost se adaptuje v průběhu evoluce na obtížnost úlohy.

3.1 Kódování prediktorů fitness

Při evoluci se souběžným učením mají jedinci tzv. plastický fenotyp. Jako plasticita se označuje schopnost jedince přizpůsobit se pomocí učení na své prostředí – jeho fenotyp pak nezáleží jen na genotypu, ale také na okolních vlivech [1]. U prediktorů s plastickým fenotypem se může jejich velikost adaptovat na složitost řešeného problému a stav evoluce. Plasticita je dosažena tím, že fenotyp prediktoru netvoří všechny ukazatele obsažené v genotypu, ale pouze jejich část.

Fenotyp se z genotypu tvoří postupným čtením genů od zvolené pozice. Pokud hodnota (ukazatel do množiny případů fitness) právě čteného genu není obsažena ve fenotypu, je do něj vložena, v opačném případě se gen ignoruje. Poté je přečten další gen a proces se opakuje. Během tvorby fenotypu nejsou čteny úplně všechny geny – jejich počet je určen proměnnou *UsedGenes*, která je součástí prostředí a v průběhu evoluce se mění. Pokud je při čtení dosaženo konce genotypu, ale nebyl ještě přečten potřebný počet genů, pokračuje se od jeho začátku. Příklad tvorby fenotypu při použití šesti genů z devíti ukazuje obrázek 3. Počátek čtení určuje zvláštní gen, který je modifikován speciálním mutačním operátorem. Ten k současné hodnotě genu přičítá malé náhodné číslo generované s normálním rozdělením.

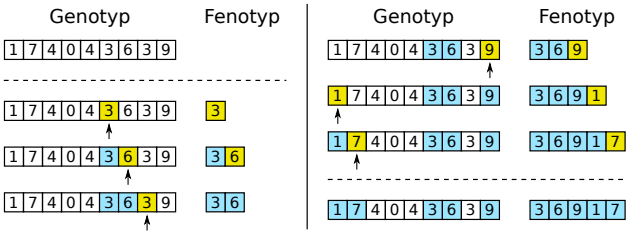
3.2 Adaptace velikosti prediktorů

Adaptace velikosti prediktoru probíhá na základě sledování rychlosti vývoje skutečné fitness kandidátních řešení. Lze očekávat, že jejich populace prochází obdobími, kdy je celková schopnost adaptace vyšší (a fitness populace stoupá) a obdobími, kdy se adaptovat příliš nedokáže (a fitness se nemění). Důležitý je i směr změny fitness, protože je pro ohodnocení jedinců v CGP používána predikovaná fitness a skutečná fitness nejlepšího jedince může i klesat. Rychlost je možné například určit jako podíl změny fitness nejlepšího jedince v populaci a počtu generací, které uběhly od minulé změny:

$$v = \frac{\Delta f_{exact}}{\Delta G} \tag{3}$$

U příliš krátkých prediktorů (v extrémním případě to může být i jen jeden případ fitness) hrozí, že se velmi dobře adaptují na kandidátní řešení uložená v archivu a predikce u jiných filtrů bude velmi nepřesná. Proto se sleduje i nepřesnost predikce a pokud překročí stanovenou mez, prediktory se prodlouží. Nepřesnost prediktoru je vypočtena při každé změně rodiče v CGP jako poměr mezi predikovanou a skutečnou hodnotou fitness nejlepšího filtru v populaci:

$$I = \frac{f_{predicted}}{f_{exact}} \tag{4}$$



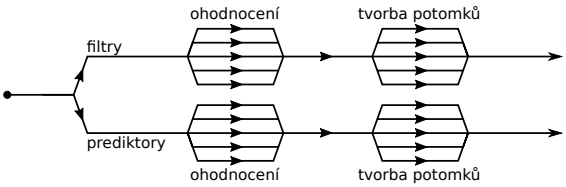
Obrázek 3. Postup konstrukce fenotypu prediktoru. Šipka označuje právě přečtený gen.

Vždy po vložení kandidátního programu do archivu je zvolen koeficient, kterým se vynásobí současná hodnota proměnné *UsedGenes*, čímž se změní velikost fenotypů prediktorů. Pravidla pro určení tohoto koeficientu jsou následující: 1) Pokud je nepřesnost predikce příliš vysoká, prediktory se prodlouží. 2) Pokud se fitness nemění ($v \approx 0$), evoluce pravděpodobně uvázla v lokálním optimu, prediktory se zkrátí, což může pomoci se z tohoto optima posunout dále. 3) Pokud fitness klesá ($v < 0$), evoluce pravděpodobně opouští lokální optimum a mírné zkrácení prediktoru může pomoci postup urychlit. 4) Pokud fitness roste ($v > 0$), prediktory se prodlouží a tím se zpřesní predikce. Rozlišuje se mezi „pomalým“ a „rychlým“ růstem.

4. Implementace

Kvůli velkému množství plánovaných experimentů byl kladen důraz především na co nejrychlejší běh programu. Implementován byl proto v jazyce C, ve kterém lze snadno pracovat i s assemblerem a rozšiřujícími instrukčními sadami procesoru.

Paralelizace je řešena pomocí direktiv překladače definovaných ve standardu OpenMP[1] pro paralelní programování se sdílenou pamětí. Rozdělení do paralelních oblastí znázorňuje obrázek 4. Při koevoluci je program rozdělen do dvou paralelních sekcí, ve kterých se vyvíjí jednotlivé populace. Ve sdílené paměti se nachází především nejlepší prediktor a archiv kandidátních programů. Paralelní sekce populací se navíc dělí na další vlákna při ohodnocování populací a při tvorbě potomků tak, že jsou jedinci zpracovávání paralelně.



Obrázek 4. Schéma paralelizace koevoluce.

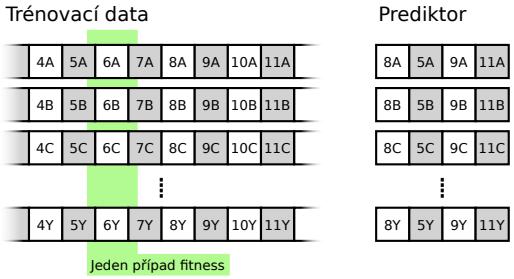
Největší důraz byl kladen na optimalizaci rychlosti výpočtu výstupních hodnot kandidátních programů, což je časově nejnáročnější část výpočtu. Ta je implementována třemi způsoby. V nejjednodušším z nich se každý výstupní pixel počítá zvlášť a pro zpracování celého obrázku o rozměrech 256×256 pixelů je třeba 65 536 průchodů filtrem. Ve druhém případě se využívá instrukční sady SSE2, které přináší vektorové instrukce nad 128bitovými registry. Díky tomu lze zpracovat 16 pixelů najednou. Nejrychlejší implementace je založená na prozatím nejnovější instrukční sadě AVX2[2], ve které jsou všechny instrukce SSE rozšířeny o podporu 256bitových registrů a lze tak počítat hodnoty 32 pixelů zároveň. Pro zpracování zmíněného obrázku pak stačí jen 2 048 průchodů filtrem.

Bohužel nelze využít starší a rozšířenější instrukční sadu AVX, protože neobsahuje instrukce pro 256bitové vektory celých čísel. V budoucnu je možné program rozšířit o implementaci s 512bitovými instrukcemi z připravované instrukční sady AVX-512[3].

Vektorizace výpočtu byla implementována pomocí tzv. intrinsic funkcí. Ty jsou překladačem přeloženy na konkrétní instrukci, například volání intrinsic funkce `_mm_add_epi8` se přeloží jako instrukce `PADDB`.

Aby byl výpočet pomocí vektorových instrukcí efektivní, je potřeba upravit strukturu dat v paměti. U sériového výpočtu je výhodné trénovací data ukládat jako jedno pole všech devítiokolí. U výpočtu pomocí SSE2 nebo AVX2 je ale třeba do registrů zkopírovat vektor 16 či 32 hodnot ze stejné pozice v devítiokolí – zde je lepší mít trénovací data uložena jako strukturu polí, z nichž každé obsahuje pixely z jedné pozice devítiokolí, resp. požadované výstupy programu.

Při ohodnocování kandidátních filtrů pomocí prediktoru jsou vstupní hodnoty náhodně rozmístěny v paměti. Proto se pro každý nový prediktor a po změně fenotypu prediktoru nejprve zkopírují vstupní data do pomocných polí tak, aby byly vybrané případy fitness bezprostředně za sebou. Struktura trénovacích dat a ukázka takto připraveného prediktoru je na obrázku 5.



Obrázek 5. Struktura trénovacích dat a prediktoru s fenotypem $\langle 8,5,9,11 \rangle$ pro výpočet instrukcemi SSE2/AVX2. Písmeno Y označuje požadovaný výstup, ostatní písmena pozice v devítiokolí.

Vektorová implementace navíc využívá toho, že v experimentech měla mřížka CGP vždy 4 řádky a l-back byl roven jedné. Aby se zmenšil počet přístupů do paměti, neukládají se výstupy funkčních bloků z předchozího sloupce do paměti, ale jsou pro ně vyhrazeny čtyři registry SSE/AVX. Další čtyři registry slouží pro výstupy bloků v právě počítaném sloupci.

Přeložený program obsahuje všechny tři implementace. Která z nich se použije při výpočtu, závisí na

[1] http://openmp.org/

[2] Poprvé představena v procesorech Intel Haswell v roce 2013.

[3] První procesory s podporou AVX-512 se objeví v roce 2015.

tom, jaké instrukční sady podporuje počítač, na kterém je program spuštěn. To se zjišťuje pomocí instrukce `CPUID`. Pokud se používají instrukce ze sady SSE2, je k dosažení stejného počtu generací CGP bez koevoluce potřeba přibližně desetina času oproti základní implementaci (pouze s OpenMP), v případě AVX2 je zrychlení zhruba šestnáctinásobné.

5. Experimentální vyhodnocení

Navržený algoritmus byl experimentálně ověřen na úloze návrhu obrazových filtrů. Cílem je nalézt vhodný filtr pro rekonstrukci obrázků poškozených impulzním šumem. U tohoto typu šumu mají poškozené pixely náhodnou hodnotu. Při experimentování se pracovalo také s šumem typu sůl a pepř, u kterého mají poškozené pixely buď minimální nebo maximální možnou hodnotu. V obou případech mohou být příčinou vadné body ve snímači v kameře, nefunkční paměťové buňky nebo chyby při přenosu dat. Tyto typy šumu jsou charakterizovány svou intenzitou – poměrným počtem poškozených pixelů.

5.1 Nastavení experimentů

Nastavení kartézského genetického programování a genetického algoritmu vychází z článku [6]. Mřížka CGP má velikost 8×4, programy mají 9 vstupů (devítiokolí zpracovávaného pixelu) a 1 výstup (nová hodnota pixelu), parametr l-back je roven jedné. V populaci je 8 jedinců, potomci jsou tvořeni mutací 1 až 5 genů.

U prediktorů fitness se v populaci vyvíjí 32 jedinců. Novou generaci tvoří 8 nejlepších jedinců, 16 jedinců vzniklých jednobodovým křížením a mutací až 5 % genů a 8 náhodných jedinců.

Parametry pro adaptaci velikost prediktorů byly určeny experimentálně. Celkem bylo spuštěno přes 170 000 nezávislých běhů programu, na jejichž základě byly zvoleny parametry pravidel změny velikosti uvedené v tabulce 1. Počáteční velikost prediktorů (hodnota *UsedGenes*) je 3 % případů fitness, minimální a maximální velikost je bez omezení. Při tvorbě fenotypu prediktoru se genotyp čte vždy od jeho počátku.

Trénovací data pochází z webu ImageProcessingPlace.com, kde se nacházejí databáze obrázků používaných komunitou zabývající se zpracováním obrazu.

Dosažené výsledky byly porovnány se standardním CGP a také s koevolučním CGP s prediktory fitness o délce 0,5 %, 1 %, 2 %, 3 %, 4 %, 5 %, 10 %, 15 %, 20 % a 25 %. Výsledky všech experimentů jsou souhrnem ze 100 nezávislých běhů pro každé nastavení po uplynutí 30 000 generací CGP.

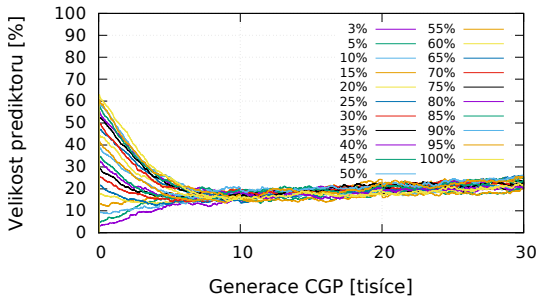
Výpočty probíhaly na superpočítači Anselm, jehož každý výpočetní uzel obsahuje dva osmijádrové procesory Intel Xeon E5-2665 (2,4 až 3,1 GHz s technologií Turbo Boost) nebo Intel Xeon E5-2470 (2,3 až 3,1 GHz s technologií Turbo Boost) a alespoň 64 GB operační paměti. Tyto procesory mají starší architekturu Sandy Bridge a nepodporují instrukční sadu AVX2.

Tabulka 1. Pravidla pro změnu proměnné *UsedGenes*. Při splnění více podmínek se uplatní první z nich.

podmínka	nová hodnota *UsedGenes*
1. $I > 1,2$	$UsedGenes \cdot 2$
2. $\lvert v \rvert \leq 0,001$	$UsedGenes \cdot 0,93$
3. $v < 0$	$UsedGenes \cdot 0,96$
4. $0 < v \leq 0,1$	$UsedGenes \cdot 1,07$
5. $v > 0,1$	beze změny

5.2 Schopnost adaptace velikosti prediktoru

Ukazuje se, že nezávisle na jejím počátečním nastavení, konverguje velikost prediktoru ke stejné hodnotě, která se liší úlohu od úlohy. Příklad průběhu velikosti u 5% šumu typu sůl a pepř je na obrázku 6. Také kvalita nalezených filtrů je srovnatelná bez ohledu na nastavení. Je tedy vhodné začínat s menšími prediktory, protože výpočet $f_{predicted}$ je pak rychlejší.



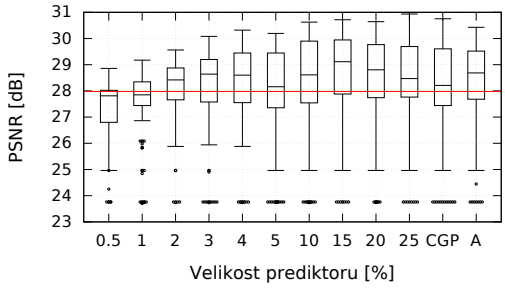
Obrázek 6. Vývoj velikosti prediktorů fitness při různé počáteční velikosti u 5% šumu sůl a pepř.

5.3 Kvalita filtrů a doba běhu

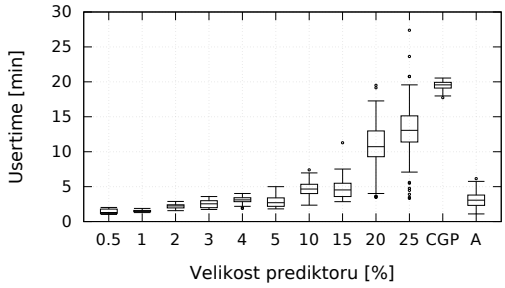
Cílem experimentů bylo srovnání navrženého algoritmu CGP s adaptivními prediktory fitness (FP_{ADAPT}) s koevolučním CGP s prediktory pevné délky (FP_{FIX}) a s CGP bez koevoluce (CGP_{STD}) s ohledem na kvalitu nalezených obrazových filtrů a na délku běhu evoluce. Experimenty byly provedeny na obrázcích se šumem o intenzitě 10 až 80 % s krokem po 10 %.

Jak je vidět na příkladu 10% impulzního šumu na obrázku 7, je kvalita filtrů nalezená pomocí FP_{ADAPT} srovnatelná s filtry získanými CGP_{STD} a FP_{FIX} s prediktory o velikosti 10 % a více. Co se týče potřebného času, je v tomto případě FP_{ADAPT} v průměru 6,26krát rychlejší než CGP_{STD}. Také je srovnatelně rychlý jako při použití FP_{FIX} s 3–5% prediktorem, kdy ale mají výsledné filtry nižší kvalitu než u FP_{ADAPT}.

Podobné výsledky byly dosaženy i pro jiné intenzity šumu. Konkrétní hodnoty jsou uvedeny v tabulce 2 a některé testovací obrázky po zpracování nejlepším nalezeným filtrem jsou na obrázku 8. Ve všech případech je kvalita filtrů získaných pomocí FP_{ADAPT} srovnatelná při kratší nebo přinejhorším podobné době běhu. V průměru bylo dosaženo 8,39násobného zrychlení oproti CGP_{STD}.



(a) Kvalita filtrů (červeně průměr PSNR všech filtrů)



(b) Čas běhu (součet všech vláken)

Obrázek 7. Srovnání CGP, koevolučního CGP s prediktory fitness různých velikostí a s prediktory fitness s adaptivní velikostí (A) pro 10% impulzní šum po 30 000 generacích.

6. Závěr

V tomto článku byl představen nový způsob kódování prediktorů fitness v koevolučním CGP, který umožňuje tvořit ze stejného genotypu různé fenotypy. Díky takto získaně plasticitě lze průběžně měnit jejich velikost a reagovat tak na průběh evoluce kandidátních řešení a na složitost řešeného problému. Z provedených experimentů na úloze návrhu obrazových filtrů vyplývá, že kvalita získaných filtrů se příliš neliší od výsledků koevoluce s prediktory pevné délky a také od výsledků standardního CGP. Oproti standardnímu CGP bylo dosaženo v průměru 8,39násobného zrychlení. Také odpadá nutnost časově náročného experimentálního hledání nejvýhodnější velikosti prediktorů pro danou úlohu, což je jedna z nevýhod koevoluce.

V dalším pokračování této práce bude představený koncept aplikován v úloze návrhu složitých kombinačních obvodů.

Poděkování

Děkuji Ing. Michaele Šikulové nejen za cenné rady, náměty a nápady, ale především za její trpělivost a za veškerý čas, který mi věnovala v souvislosti s řešením této a diplomové práce.

Tato práce byla podporována projekty Vysokého učení technického v Brně FIT-S-14-2297 a Centra excelence IT4Innovations CZ.1.05/1.1.00/02.0070.

Literatura

[1] ELLEFSEN, K. O.: Balancing the Costs and Benefits of Learning Ability. In: *Advances in Artificial Life, ECAL 2013*, Cambridge (USA): MIT Press, 2013, ISBN 978-0-262-31709-2, s. 292–299.

[2] MILLER, J. F.: Cartesian Genetic Programming. In: *Cartesian Genetic Programming*, Springer-Verlag Berlin Heidelberg, 2011, s. 17–34. ISBN 978-3-642-17309-7.

[3] SCHMIDT, M.; LIPSON, H.: Coevolution of Fitness Predictors. *IEEE Transactions on Evolutionary Computation*. 2008, ročník 12, č. 6: s. 736–749, ISSN 1089-778X.

[4] SEKANINA, L.; HARDING, S. L.; BANZHAF, W.; aj.: Image Processing and CGP. In: *Cartesian Genetic Programming*, Springer-Verlag Berlin Heidelberg, 2011, s. 181–214. ISBN 978-3-642-17309-7.

[5] ŠIKULOVÁ, M.; HULVA, J.; SEKANINA, L.: Indirectly Encoded Fitness Predictors Coevolved with Cartesian Programs. In: *Proc. of the 18th European Conference on Genetic Programming*, Springer, 2015, s. 113–125. Lecture Notes in Computer Science č. 9025. ISBN 978-3-319-16500-4.

[6] ŠIKULOVÁ, M.; SEKANINA, L.: Acceleration of evolutionary image filter design using coevolution in cartesian GP. In: *Parallel Problem Solving from Nature-PPSN XII*, Springer, 2012, s. 163–172. Lecture Notes in Computer Science č. 7491. ISBN 978-3-642-32937-1.

[7] ŠIKULOVÁ, M.; SEKANINA, L.: Coevolution in cartesian genetic programming. In: *Genetic Programming*, Springer, 2012, s. 182–193. Lecture Notes in Computer Science č. 7244. ISBN 978-3-642-29138-8.

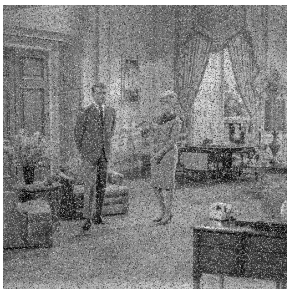
[8] VANNESCHI, L.; POLI, R.: Genetic Programming – Introduction, Applications, Theory and Open Issues. In: *Handbook of Natural Computing*, Springer, 2011, s. 710–734. ISBN 978-3-540-92909-3.

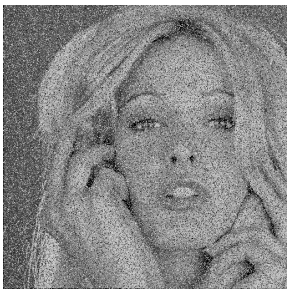
Tabulka 2. Srovnání standardního CGP, CGP s koevolucí s prediktory různých délek a s prediktory s adaptivní velikostí po 30 000 generacích. Řešenou úlohou je filtr pro impulzní šum různé intenzity. U FP_{FIX} byly testovány délky 0,5, 1, 2, 3, 4, 5, 10, 15, 20 a 25 procent, v tabulce je jen výsledek s nejlepším průměrným PSNR. U FP_{ADAPT} je uvedena průměrná délka prediktoru.

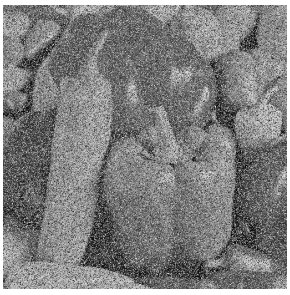
Intenzita			CGP_{STD}	FP_{FIX}	FP_{ADAPT}
10 %	Prediktor [%]		–	15	7,59
	PSNR [dB]	Min.	23,76	23,76	23,76
		Průměr	27,91	28,61	28,20
		Max.	30,75	30,72	30,43
	Čas [min]	Min.	17,73	2,85	1,10
		Průměr	19,46	4,72	3,11
		Max.	20,55	11,28	6,15
20 %	Prediktor [%]		–	25	4,08
	PSNR [dB]	Min.	20,91	20,91	20,91
		Průměr	24,80	25,18	24,82
		Max.	27,31	27,57	27,07
	Čas [min]	Min.	16,97	3,67	1,18
		Průměr	19,42	9,30	2,66
		Max.	20,73	24,80	4,00
30 %	Prediktor [%]		–	15	2,94
	PSNR [dB]	Min.	19,11	19,11	18,98
		Průměr	22,37	22,56	22,35
		Max.	24,54	24,42	24,15
	Čas [min]	Min.	14,83	5,28	1,12
		Průměr	19,33	9,49	2,49
		Max.	20,72	23,78	4,22
40 %	Prediktor [%]		–	15	1,65
	PSNR [dB]	Min.	17,66	17,38	17,22
		Průměr	20,11	20,21	19,79
		Max.	22,25	21,81	21,53
	Čas [min]	Min.	17,43	2,58	1,02
		Průměr	19,31	6,53	2,45
		Max.	20,62	22,32	3,80

Intenzita			CGP_{STD}	FP_{FIX}	FP_{ADAPT}
50 %	Prediktor [%]		–	4	1,22
	PSNR [dB]	Min.	16,49	16,48	15,75
		Průměr	18,20	18,35	18,04
		Max.	19,40	19,55	19,02
	Čas [min]	Min.	18,00	2,13	1,53
		Průměr	19,49	4,54	2,64
		Max.	20,62	7,53	4,12
60 %	Prediktor [%]		–	20	0,80
	PSNR [dB]	Min.	15,08	15,48	15,47
		Průměr	16,71	16,81	16,52
		Max.	17,45	17,48	17,04
	Čas [min]	Min.	17,50	5,00	0,85
		Průměr	19,44	11,98	2,92
		Max.	20,43	47,73	5,27
70 %	Prediktor [%]		–	20	0,62
	PSNR [dB]	Min.	14,66	14,64	14,52
		Průměr	15,64	15,65	15,43
		Max.	16,08	16,04	15,86
	Čas [min]	Min.	17,00	5,28	0,85
		Průměr	19,41	18,22	3,01
		Max.	20,43	56,27	4,98
80 %	Prediktor [%]		–	10	0,65
	PSNR [dB]	Min.	14,52	14,52	14,52
		Průměr	14,85	14,85	14,70
		Max.	15,11	15,15	15,06
	Čas [min]	Min.	16,52	4,72	0,80
		Průměr	19,41	11,24	1,96
		Max.	20,57	24,85	3,40


(a) Obrázek 1 (20% šum)


(b) Obrázek 2 (30% šum)


(c) Obrázek 3 (40% šum)


(d) Obrázek 4 (50% šum)


(e) Filtrovaný obrázek 1


(f) Filtrovaný obrázek 2


(g) Filtrovaný obrázek 3


(h) Filtrovaný obrázek 4

Obrázek 8. Testovací obrázky filtrované nejlepšími filtry (jeden krok filtrace) nalezenými pomocí koevoluce s adaptivními prediktory fitness.