BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2018
http://excel.fit.vutbr.cz

Symbolická regrese: Jak získat kvalitní a současně kompaktní řešení?

Bc. Ondřej Končal

Abstrakt

Geometrické sémantické genetické programování (GSGP) dosahuje kvalitních výsledků při popisu dat složitými matematickými modely. Cenou za přesný popis je ale výsledná velikost řešení. Tento článek se zabývá převodem řešení získaného GSGP na instanci kartézského genetického programování (CGP) a jeho následnou optimalizací. CGP dokáže dobře redukovat velikost již vzniklých řešení a kombinace těchto dvou metod má tak potenciál vytvořit kvalitní a zároveň malý model popisující vstupní data. Úspěšným principem redukce je podstromové kartézské genetické programování (SCGP), které je představeno v tomto článku. Využívá možnosti rozdělení řešení na podstromy a následně je upravuje. Na všech testovaných úlohách z oblasti symbolické regrese se podařilo dosáhnout znatelného zmenšení řešení a pouze u jedné úloh ze 4 došlo k přetrénování. Kombinace GSGP a SCGP tak má potenciál vytvořit přesnější model než genetické programování nebo CGP za řádově stejnou dobu. Velikost řešení je znatelně menší než samotný výstup GSGP.

Klíčová slova: geometrické sémantické genetické programování — kartézské genetické programování — symbolická regrese — evoluční algoritmus

Přiložené materiály: Naměřená data a vyvinutá řešení

*xkonca00@stud.fit.vutbr.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

1. Úvod

Evoluční algoritmy jsou dnes již zaběhlým a účinným nástrojem nejen pro optimalizaci, ale některé z nich jsou schopny automaticky vytvářet spustitelné struktury. Takovým algoritmem je genetické programování [1], které dokáže vytvářet a optimalizovat jak počítačové programy, tak i rovnice a obvody. Standardní genetické operátory v genetickém programování pracují na úrovni syntaxe a genotypu, ale cílem je úprava fenotypu. Cílenou změnu fenotypu a s tím i sémantiky přináší geometrické sémantické operátory a jejich použití v geometrickém sémantickém genetickém programování
(GSGP) [2]. Cenou za kvalitní řešení a téměř žádné přetrénování je velikost výsledného řešení – počet použitých uzlů grafu. Každá aplikace geometrického sémantického operátoru jedince zvětší, ale díky speciálnímu přístupu je evaluace rychlá. Nicméně rekonstrukce výsledného řešení je časově a prostorově náročná, stejně tak i použití tohoto řešení. S možností zmenšení již existujících řešení přichází kartézské genetické programování (CGP) [3], které v tomto směru již prokázalo svoji funkčnost.

Hlavním cílem tohoto článku je návrh algoritmu, který efektivně zmenší velikost řešení získaného pomocí GSGP. Je zde představen jak algoritmus převodu řešení získaného pomocí GSGP na instanci CGP, tak i speciální přístup využití CGP nazvaný podstromové CGP (SCGP), které pracuje vždy pouze se sémanticky spjatou částí chromozomu. Experimenty jsou provedeny na reálných farmakokinetických datech.

Článek je uspořádán následovně: Sekce 2 se zabývá stručným přehledem symbolické regrese a GSGP, sekce 3 přehledem CGP. V sekci 4 je představen algoritmus převodu výstupu GSGP na reprezentaci CGP. V sekci 5 je představeno SCGP. Výsledky experimentů a data s nastavením k nim použité jsou popsány v sekci 6. Nakonec jsou shrnuty výsledky experimentů a navrženy možnosti budoucí práce v oblasti optimalizace výstupu geometrického sémantického genetického

programování pomocí kartézského genetického programování.

2. Geometrické sémantické genetické programování

Jednou z klasických úloh, ve které se genetické programování uplatňuje, je symbolická regrese [1]. Jedná se o nalezení výrazu
$E : \mathbb{R}^n \to \mathbb{R}$ co nejlépe popisujícího vstupní data tvořená dvojicemi (v_i, y), kde $v_i \in \mathbb{R}^n$ a $y \in \mathbb{R}$. Často je využíván výpočet hodnoty fitness (kvality řešení) f pomocí průměrné odchylky:

$$f = \frac{\sum_{j=1}^{N} |y_j^{GP} - y_j|}{N} \tag{1}$$

kde y_j^{GP} značí výstupy kandidátního programu, které jsou porovnávané s výstupy trénovací množiny y_j o velikosti N. Jedná se o úlohu minimalizace hodnoty fitness. Problém symbolické regrese lze řešit i pomocí GSGP.

GSGP je forma genetického programování, která upravuje jedince přímo na úrovni sémantiky za pomoci geometrických sémantických operátorů. Těmi jsou geometrická sémantická mutace (GSM) a geometrické sémantické křížení (GSC):

$$\text{GSC: } T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2) \tag{2}$$

$$\text{GSM: } T_M = T + ms \cdot (T_{R1} - T_{R2}) \tag{3}$$

kde $T, T_1, T_2 : \mathbb{R}^n \to \mathbb{R}$ jsou rodiče, $T_R, T_{R1}, T_{R2} : \mathbb{R}^n \to \mathbb{R}$ jsou náhodné reálné funkce s oborem hodnot $\langle 0, 1 \rangle$ a *ms* je reálná hodnota označovaná jako *mutation step*, která udává velikost mutace. Pro upravení náhodně vygenerovaných funkcí tak, aby vracely hodnoty v rozsahu $\langle 0, 1 \rangle$, je použita speciální logistická funkce sigmoida:

$$T_R = (1 + e^{-T_{rand}})^{-1} \tag{4}$$

kde T_{rand} je funkce vracející neomezenou hodnotu. Tyto operace tak stále zvětšují jedince, na které jsou aplikovány. Jako poslední operace je používána replikace, která přenese jedince z minulé generace nezměněného.

Implementace GSGP v C++ představená v [2] využívá namísto generování stále nových náhodných stromů a následného fyzického spojení stromů do jednoho metodu záznamů s odkazy. Na začátku je vygenerována počáteční populace a k ní multimnožina náhodně vygenerovaných stromů. Všichni jedinci jsou pak tvořeni pouze odkazy na tyto stromy a záznamy pak vypadají $(crossover, \&T_1, \&T_2, \&T_R)$ nebo $(mutation, \&T, \&T_{R1}, \&T_{R2}, ms)$. Ohodnocení jedinců lze vytvořit vždy z ohodnocení podstromů, z kterých se skládají, což značně urychlí evaluaci. Pro získání výsledku je potřeba provést zpětnou rekonstrukci, která je časově a prostorově náročná. Výsledky experimentů ukázaly, že GSGP si dokáže velmi dobře poradit s přetrénováním a řešení tak zůstává stále dostatečně obecné (viz [4]).

Pro účely této práce je využita výše zmíněná implementace GSGP od Maura Castelli dostupná na stránkách http://gsgp.sourceforge.net/.

3. Kartézské genetické programování

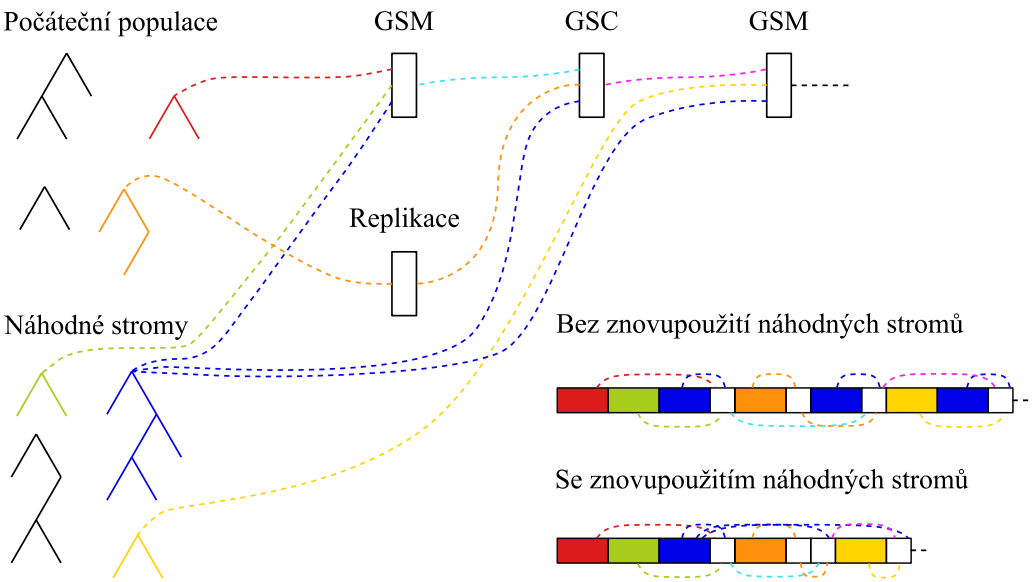
V CGP jsou reprezentovány spustitelné struktury pomocí acyklického grafu ve formě dvojrozměrné mřížky dané velikosti [3]. Přirozeně tak dokáže pracovat s obvody a matematickými funkcemi a zároveň zabraňuje i vzniku *bloatu*, tzn. zvětšování stromu bez změny hodnoty fitness, který se u standardního genetického programování vyskytuje. Jediným genetickým operátorem používaným v CGP je mutace, která má jak explorativní, tak exploitativní funkci. Je tomu tak díky možnosti použít vstupy a vnitřní uzly i vícekrát (nebo vůbec), což vede na několik druhů redundance. To umožňuje znovupoužití částí grafu a možnost zakódování většího fenotypu. Uzly, které jsou použity pro tvorbu fenotypu (jsou propojeny s výstupem), se nazývají aktivní. Evoluce používá strategii $(1 + \lambda)$, kde počet potomků $\lambda \in \langle 4, 9 \rangle$.

Využívá se jak k návrhu nových, tak k optimalizaci již vzniklých řešení, které může zmenšovat nebo zrychlovat.

4. Převod výstupu GSGP na instanci CGP

Prvním krokem je převod výstupu GSGP do CGP, aby bylo možno jej dále zpracovat. Tato část je využitelná i mimo SCGP a proto je zde představena samostatně. Jak zde bylo již řečeno, řešení je uloženo jakožto počáteční populace stromů a multimnožina stromů náhodně vygenerovaných na začátku běhu algoritmu. Jednotlivé generace obsahují pouze záznamy o tom, které záznamy z minulé generace, případně počáteční a náhodné stromy jsou spojeny. Zpětným průchodem se všechny použité stromy a záznamy označí. Náhodné stromy mohou být použity i vícekrát, proto je potřeba tuto skutečnost zakomponovat do návrhu a jsou zde představeny dva přístupy – vložení reprezentace náhodného stromu do chromozomu znova, nebo odkázání se na již uloženou reprezentaci.

Algoritmus postupně prochází přes všechny generace a každý záznam, který je označen jako použitý pro tvorbu výsledku, se přidá do chromozomu. Pokud



Obrázek 1. Ilustrace převodu jedince v GSGP na jedince v CGP dvěma způsoby

se v záznamu objeví odkaz na náhodný strom nebo jedince z počáteční populace, je jeho první výskyt vždy přiveden na postfixovou notaci a následně na odpovídající uzly CGP. V případě opakovaného výskytu stromu je možno vytvořit strom znovu, nebo se připojit na uzel CGP se stromem již vytvořeným (obr. 1) (v druhém případě je potřeba ukládat indexy uzlů, které obsahují vrcholy stromů a vždy testovat, zda byl už strom použit). Při použití GSM, GSC a replikace je proveden odkaz vždy na uzly vytvořené v minulé generaci, případně na náhodné stromy (uzly), se kterými jsou spojeny (u GSM a GSC dodány další uzly propojující tři podstromy).

5. Podstromové kartézské genetické programování

Hlavní částí tohoto článku je návrh algoritmu využívajícího CGP ke zmenšení velikosti řešení získaného pomocí GSGP. Tento přístup je zde nazván podstromové CGP proto, že pracuje odděleně s podstromy, které byly postupně přidávány v jednotlivých generacích GSGP. Řešení získané z GSGP musí splňovat následující podmínky:

1. Nepoužívá se geometrické sémantické křížení (pouze GSM a replikace)
2. Převod do CGP je realizován bez znovupoužití náhodných stromů

Hlavní myšlenka je v rozdělení chromozomu na menší a sémanticky spjaté chromozomy obsahujících jednotlivé GSM a původní strom. Díky asociativitě a komutativitě sčítání lze upravovat a vyhodnocovat jednotlivé chromozomy nezávisle, rychleji a lokálněji. Nad každým je tak možné spustit samostatný běh CGP.

Nejprve je výsledek GSGP nepoužívajícího GSC převeden do chromozomu CGP bez znovupoužití náhodných stromů. Následně je chromozom rozdělen na jednotlivé GSM chromozomy tak, jak byly v generacích přidávány. Pro každý chromozom je vytvořen vektor odezev na vstupní vektory, což způsobí značné zrychlení výpočtu hodnoty fitness v pozdější fázi algoritmu. Následuje cyklus, ve kterém se postupně spouští CGP nad každým chromozomem. Pro výpočet hodnoty fitness daného chromozomu je jakožto požadovaný výstup použit výstup z trénovacích dat snížený o odezvy všech ostatních chromozomů. Dále probíhá klasické CGP s parametry $(generations, population\_size, mutation\_rate, selection\_method)$. V posledním kroku jsou odstraněny uzly, které nejsou aktivní a je přepočítán vektor odezev na vstupy pro evolvovaného jedince. Po evoluci chromozomů následuje sjednocení chromozomů po dvou uzlem s operací součtu. Chromozomy jsou vybírány dle zadané strategie *subtree_select_method*. V poslední fázi dojde opět k výpočtu odezev všech chromozomů na vstupní vektory. Cyklus evoluce chromozomů za pomoci CGP a následné spojování probíhá tak dlouho, dokud nezbude jediný chromozom, který prošel evolucí.

6. Experimenty

6.1 Nastavení

Parametry použité pro nastavení GSGP při těchto experimentech jsou následující:

- Velikost populace: 2000
- Počet generací: 1000 (případně 300 pro *P3D*)
- Počet náhodných stromů: 500
- Metoda tvorby stromů: Ramped Half-and-Half
- Maximální hloubka stromu: 8
- Pravděpodobnost křížení: 0

Pravděpodobnost mutace: 0,9
Velikost turnaje: 4

Bylo provedeno 40 nezávislých běhů pro vybraná data, kdy řešení s nejmenší hodnotou fitness bylo vybráno jakožto vstup SCGP. Použité parametry SCGP jsou:

Velikost populace: 8
Počet generací: 50
Pravděpodobnost mutace: 0,08
Metoda selekce: nejkratší jedinec s fitness lepší nebo stejně dobrou jako byla počáteční
Metoda selekce podstromů: náhodný výběr

Pro každý vybraný výstup GSGP bylo provedeno 20 nezávislých běhů.

6.2 Trénovací a testovací data

Testování proběhlo na 4 úlohách z oblasti farmakokinetiky. Jedná se o data popisující lidskou orální biodostupnost (*bioav*), úroveň vazby plazmatických bílkovin (*PPB*), medián smrtící dávky – toxicitu (*Tox*) a 3D strukturu proteinu (*P3D*). První 3 datasety jsou důkladně rozebrány v [5]. Rozměry datasetů jsou popsány v tabulce 1 (funkce mají jeden výstup – poslední sloupec). Každý dataset je 30-ti různými způsoby rozdělen na trénovací a testovací část v poměru 70:30 a pro experimenty bylo náhodně vybráno jedno rozdělení. Všechna rozdělení jsou nakonec použita pro otestování přetrénování.

	vstupní parametry (sloupce-1)	záznamy (řádky)
bioav	241	359
PPB	628	131
Tox	626	234
P3D	9	45730

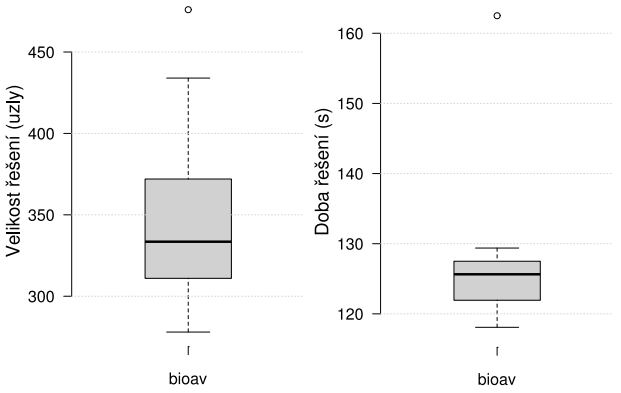
Tabulka 1. Parametry použitých datasetů

6.3 Výsledky

Základní údaje o výsledcích budou prezentovány vždy v trojici (počet aktivních uzlů; trénovací fitness; testovací fitness). Grafy jsou vytvořeny ze všech běhů SCGP. Nejlepší jedinec je vybrán nejen jako nejmenší, ale zároveň musí splňovat i testovací fitness, která není kontrolována programem. Čím menší (lepší) řešení bude, tím rychleji bude zpravidla evoluce ukončena.

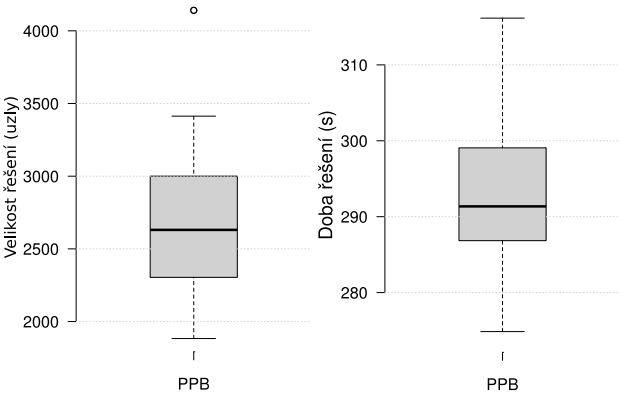
Použití klasického CGP jak přímo na vstupní data nebo na výstup GSGP se ukázalo jako neefektivní. Výsledky byly vždy horší než u samotného GSGP, případně SCGP.

Nejlepší výstup GSGP pro *bioav* měl hodnoty (22285; 19,455; 26,938). Výsledek 20-ti běhů SCGP je zobrazen na obr. 2. Podmínku pro testovací fitness splňují 3 výsledky: (330; 19,412; 26,465), (332; 19,378; 25,839), (339; 19,443; 25,691). To značí zmenšení o 98,5% za necelých 130 sekund ve všech případech. Všechna tři řešení se ukázala stejně obecná jako výsledek získaný pomocí GSGP.



Obrázek 2. bioav – původní velikost 22285 uzlů

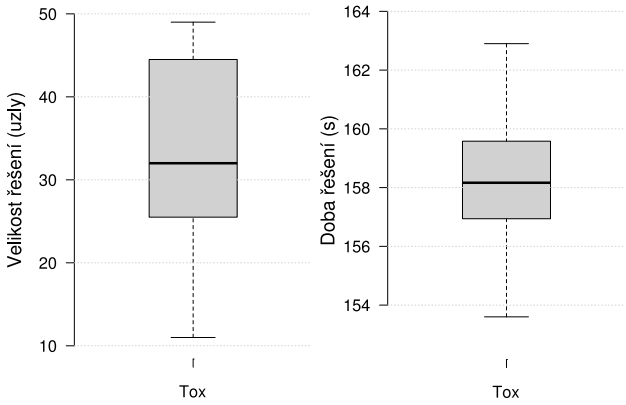
GSGP pro *PPB* data našlo řešení (22907; 4,312; 30,768). Výsledky SCGP jsou vizualizovány na obr. 3. Nalezeno bylo jediné ucházející řešení o hodnotách (2722; 4,293; 30,718). Zmenšení tak činí 88% za 295 sekund. Řešení si nedokázalo poradit s přetrénováním ani v jednom případě (kromě původního trénovacího a testovacího).



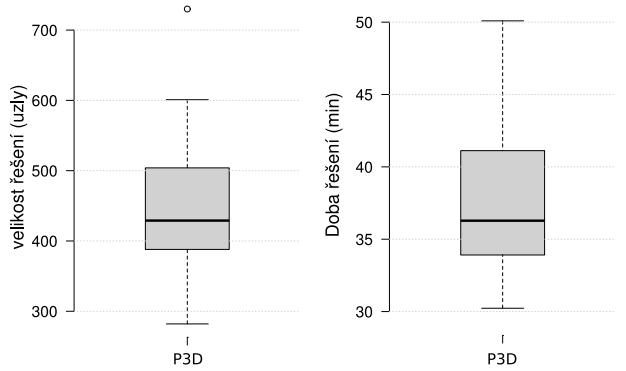
Obrázek 3. PPB – původní velikost 22907 uzlů

Výstup GSGP popistující dataset *Tox* má hodnoty (21568; 1336,956; 1495,621). Na obr. 4 je vidět naměřené hodnoty z běhů SCGP. 10 výsledků je přijatelných s velikostí 11 až 47. Zmenšení je více jak 99,5% za zhruba 160 sekund. Každé řešení si zhruba zachovalo stejnou schopnost zobecňovat jako referenční řešení poskytnuté GSGP.

Vstup SCGP pro úlohu *P3D* nabývá hodnot (5764; 3,981; 3,999) a výstup je graficky zobrazen na obr. 5. Kvůli vysokému počtu záznamů trvá evoluce GSGP pouze 300 generací. Z 20-ti běhů je 15 dostatečně kvalitních s rozsahem velikostí 282 až 730, což dává zmenšení 87,3% až 95,1% do 3000 sekund. Tři běhy nejsou v grafu zaneseny, neboť skončily výjimkou. Všechna řešení jsou podobně obecná jako výstup GSGP.



Obrázek 4. Tox – původní velikost 21568 uzlů



Obrázek 5. P3D – původní velikost 5764 uzlů

7. Závěr

V této práci byly představeny dvě metody – Převod řešení GSGP na instanci CGP a podstromové CGP. GSGP pracuje se sémantickými operátory a dokáže dobře zobecňovat, nicméně vytvořené řešení je velké. Úlohou SCGP je tento výstup zmenšit. Experimenty ukázaly, že ve třech případech (datasety *bioav*, *Tox*, *P3D*) je dosažitelné zmenšení větší jak 95% a přetrénování nenastává. Pro dataset (*PPB*) zmenšení činilo 88%, ale přetrénování se ukázalo jako neúnosné.

Další práce bude probíhat na poli experimentování s nastavováním parametrů evoluce, metodou výběru podstromů pro sjednocení a spouštění klasického CGP na již zmenšeném výsledku. Jako možnost dalšího výzkumu se nabízí navrhnout obdobu SCGP pracující i s geometrickým sémantickým křížením a také vyzkoušet SCGP na nově představenou formu mutace pro GSGP – geometrickou sémantickou mutaci s lokálním prohledáváním.

Poděkování

Tímto bych rád poděkoval svému vedoucímu prof. Ing. Lukáši Sekaninovi, Ph.D. za odborné vedení, vstřícnost při konzultacích a cenné rady, které mi pomohly tuto práci vyhotovit.

Literatura

[1] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge : Bradford Book, London : MIT Press, 1992.

[2] Mauro Castelli, Sara Silva, and Leonardo Vanneschi. A c++ framework for geometric semantic genetic programming. *Genetic Programming and Evolvable Machines*, 16(1):73–81, Mar 2015.

[3] Lukáš Sekanina, Zdeněk Vašíček, Richard Růžička, Michal Bidlo, Jiří Jaroš, and Petr Švenda. *Evoluční hardware: od automatického generování patentovatelných invencí k sebemodifikujícím se strojům*. Praha : Academia, 2009. ISBN 978-80-200-1729-1.

[4] Mauro Castelli, Leonardo Trujillo, Leonardo Vanneschi, Sara Silva, Emigdio Z-Flores, and Pierrick Legrand. Geometric semantic genetic programming with local search. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 999–1006, New York, NY, USA, 2015. ACM.

[5] Francesco Archetti, Stefano Lanzeni, Enza Messina, and Leonardo Vanneschi. Genetic programming for computational pharmacokinetics in drug discovery and development. *Genetic Programming and Evolvable Machines*, 8(4):413–432, Dec 2007.