COEVOLUTIONARY ALGORITHM FOR TEST-BASED PROBLEMS

Jiří Hulva
Master Degree Programme (3.), FIT BUT
E-mail: xhulva00@stud.fit.vutbr.cz

Supervised by: Michaela Šikulová
E-mail: isikulova@fit.vutbr.cz

Abstract: This paper deals with the usage of coevolution in the task of symbolic regression. Symbolic regression is used for obtaining mathematical formula which approximates the measured data. It can be executed by cartesian genetic programming - a method from the cathegory of evolutionary algorithms that is inspired by natural evolutionary processes. Coevolution works with multiple evolutionary processes running simultaneously and influencing each other. This paper is focused on the development of an algorithm which performs symbolic regression using coevolution on test-based problems. Use of coevolution should improve speed and accuracy of the task significantly.

Keywords: Coevolutionary algorithm, symbolic regression, cartesian genetic programming

1 ÚVOD

Mnoha vědním oborům je společná snaha matematicky popsat zkoumané přírodní jevy, objevit v naměřených údajích skryté souvislosti. Tradičně výzkumníkům nezbývalo nic jiného, než se spolehnout na své znalosti daného problému. Ze získaných dat museli umět odhadnout, jakým způsobem by mohly dané veličiny na sobě navzájem záviset. Následně pomocí některé z metod regresní analýzy dopočítali parametry tohoto vztahu. S rostoucím objemem naměřených dat a počtem zkoumaných veličin se však tento úkol stává čím dál tím složitější. Je zde proto snaha celý proces plně zautomatizovat a využít výpočetní síly počítačů. Hledání matematického vztahu závislosti naměřených veličin vyžaduje prohledávání rozsáhlých prohledávacích prostorů. K tomu je nutné využít heuristické stochastické algoritmy. Tato úloha nalezení souvislostí mezi veličinami, stejně jako parametry těchto vztahů, se nazývá symbolická regrese. Používá se při ní algoritmus genetického programování a různé jeho varianty – například kartézské genetické programování.

2 KARTÉZSKÉ GENETICKÉ PROGRAMOVÁNÍ

Kartézské genetické programování (CGP) je algoritmus inspirovaný přírodními evolučními procesy. Pracuje s populací možných řešení zkoumaného problému. Jednotlivá kandidátní řešení se hodnotí, jak dobře splňují cílové podmínky (resp. jak dobře řeší zadanou úlohu). Z úspěšnějších se vytvoří nová generace tak, že je změníme pomocí operátorů inspirovaných evoluční teorií – v případě CGP je to operátor mutace. To se opakuje tak dlouho, dokud není nalezen výsledek. Podobně jako v přírodě se tak populace postupně adaptuje na cílové podmínky. V CGP je kandidátní řešení reprezentováno jako mřížka vzájemně propojených uzlů, tvořících acyklický orientovaný graf. Každý uzel má přidělenu matematickou funkci, pomocí které ze svých vstupních hodnot vypočte hodnotu výstupní. Ohodnocení jedince probíhá pomocí trénovacích vektorů – n-tic vstupních hodnot a odpovídajících funkčních hodnot – správných výstupů. Při ohodnocování se na primární vstupy jedince přivedou vstupní hodnoty a porovná se jeho výstupní hodnota s požadovaným výstupem v trénovacím vektoru. V závislosti na tom je jedinci přidělena *fitness hodnota*, udávající jeho kvalitu.

Kartézské genetické programování (stejně jako genetické programování obecně) má i své nevýhody. Často se stává, že prohledávání uvázne v lokálním optimu, a tedy nedojde k nalezení řešení požadované kvality. Také bývá velmi časově náročné – při prohledávání prostoru možných řešení je třeba vyhodnotit miliony a více kandidátních řešení. Bylo ukázáno [1], že oba tyto problémy lze do značné míry eliminovat pomocí *koevoluce*, kdy se vyvíjí více rozdílných populací zároveň a navzájem se ovlivňují. Tímto přístupem lze evoluční proces značně urychlit. Navíc vzhledem k dynamicky se měnícím podmínkám nedochází k uváznutí prohledávání.

3 KOEVOLUČNÍ ALGORITMUS PRO ÚLOHY ZALOŽENÉ NA TESTU

Termín „úlohy založené na testu"označuje koevoluci pracující se dvěma typy populací, jejichž evoluce probíhá paralelně [2]. Kandidátní řešení jsou vyvíjena v první populaci, zatímco druhá populace slouží pro otestování kvality nalezených řešení. Pro evoluci jedinců z obou populací je použito CGP.

Jedinci z první populace – *kandidátní řešení* – se snaží nalézt řešení zadaného problému, představují hledané matematické vztahy. Jsou ohodnoceni podle toho, jak dobře jejich odezvy na trénovací vektory odpovídají očekávaným hodnotám. Výpočet fitness kandidátních řešení je na principu *skóre*. Jestliže výsledek pro daný trénovací vektor odpovídá funkční hodnotě trénovacího vektoru (s definovanou chybou σ), tak se k fitness hodnotě přičte 1, jinak 0. Čím větší má kandidátní řešení skóre, tím je kvalitnější:

$$fitness(s) = \sum_{i=1}^{n} g\big(s(x_i)\big), \text{ kde} \tag{1}$$

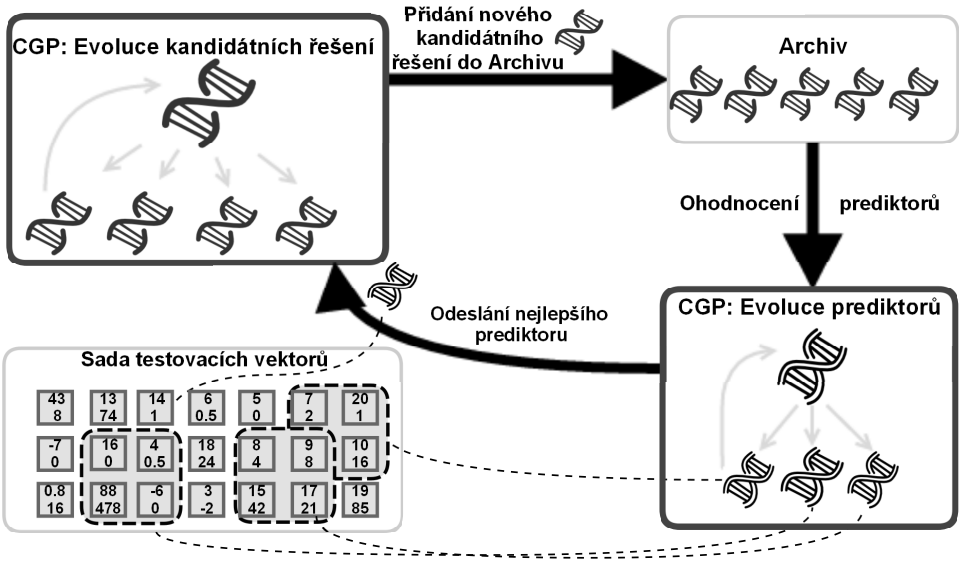
$$g(s(x_i)) = \begin{cases} 0 & \text{pro } |y_i - s(x_i)| \geq \sigma \\ 1 & \text{pro } |y_i - s(x_i)| < \sigma \end{cases} \tag{2}$$

kde s je kandidátní řešení (algebraický výraz), x_i je trénovací vstup funkce a y_i je očekávaný výstup od správně pracujícího řešení.

Jedinci z druhé populace – *prediktory* – slouží k ohodnocení jedinců z populace kandidátních řešení. Vybírají, které trénovací vektory se použijí, a které ne. V případě evoluce prediktorů je hledán matematický výraz, který mapuje prvky z uspořádané množiny trénovacích vektorů – vybírá, které prvky budou použity pro ohodnocení kandidátních řešení. Ohodnocování jedinců patří k časově nejnáročnějším částem evoluce, zvláště pokud je sada trénovacích vektorů značně rozsáhlá. Díky tomu, že se použije pouze zlomek trénovacích vektorů, se evoluce kandidátních řešení značně urychlí. Cenou za to je ovšem fakt, že takovéto ohodnocení není úplně přesné, jedná se jen o odhad. Samy prediktory se vyvíjejí tak, aby vybíraly právě ty testy z trénovací sady, které celkové ohodnocení vystihují co nejlépe. Mají k dispozici několik kandidátních řešení v tzv. *Archivu*. Na nich se testuje, jak dobře odpovídá ohodnocení pomocí prediktorů tomu skutečnému (pomocí celé sady trénovacích vektorů). V tomto případě se určí hodnota fitness podle vztahu:

$$f(p) = \frac{1}{n} \sum_{a=1}^{m} \big(fitness(a) - fitness'(a)\big)^2 \tag{3}$$

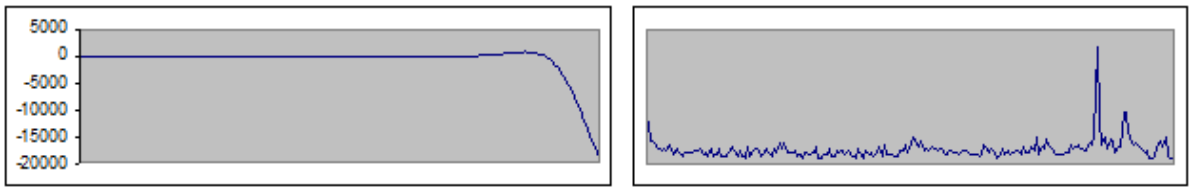
kde a jsou kandidátní řešení v archivu, $fitness$ je skutečná fitness kandidátního řešení a $fitness'$ je fitness hodnota kandidátního řešení predikovaná daným prediktorem. Čím je tato fitness menší, tím je prediktor kvalitnější. Jednou za čas se odešle nejlepší prediktor do populace kandidátních řešení, kde je použit k jejímu ohodnocování. Schéma koevoluční interakce můžete vidět na obrázku 1.



Obrázek 1: Schéma koevolučního algoritmu pro úlohu symbolické regrese.

4 CÍL PRÁCE

V dřívějších pracích pojednávajících o koevoluci a symbolické regresi, například [1], byl pro evoluci prediktorů použit jednoduchý genetický algoritmus. Jedinci byli rozsáhlé vektory indexů vybraných testů. V mé práci používám pro evoluci prediktorů kartézské genetické programování. Jedinec představuje matematický vztah, který požadované indexy testů vygeneruje. Díky tomu může evoluce prediktorů probíhat mnohem rychleji, nezávisle na velikosti trénovací sady. Také se dříve ukázalo, že podstatný vliv na efektivitu evolučního procesu má velikost trénovací podmnožiny, kterou vybere prediktor. Tuto velikost bylo dříve nutné předem nastavit. Není však možné napevno zvolit nějaký optimální kompromis mezi rychlostí evoluce a přesností, protože tato hodnota se značně liší u každé zkoumané úlohy. Použití kartézského genetického programování dovoluje, aby se velikost trénovací podmnožiny dynamicky měnila. První experimenty ukazují (Obrázek 2), že se trénovací podmnožina do určité míry dokáže adaptovat na řešenou úlohu. Experimenty probíhají nad úlohami symbolické regrese, které používá komunita zabývající se symbolickou regresí pro vyhodnocování kvality svých mechanismů.



Obrázek 2: Vlevo: graf trénovací funkce $y = e^x cos(x)$; Vpravo: oblasti vybrané prediktorem

PODĚKOVÁNÍ

Článek vznikl za podpory projektu VUT v Brně FIT-S-14-2297.

REFERENCE

[1] Šikulová, M.; Sekanina, L.: Coevolution in Cartesian Genetic Programming. *Proc. of the 15th European Conference on Genetic Programming*, LNCS 7244, Springer Verlag, 2012, ISBN 978-3-642-29138-8, s. 182–193.

[2] Popovici, E.; Bucci, A.; Wiegand, R. P.; aj.: Coevolutionary principles. *Handbook of Natural Computing*, Springer, 2012, ISBN 978-3-540-92909-3, s. 987–1033.