VYČÍSLOVÁNÍ KONSTANT PRO GENETICKÉ PROGRAMOVÁNÍ (CONSTANT EVALUATION FOR GENETIC PROGRAMMING)

Vladimír Hlaváč[1]
[1] *ČVUT FS, hlavac@fs.cvut.cz*

Abstrakt: Při použití genetického programování pro symbolickou regresi je důležitá role číselných hodnot konstant. Některé běžně používané metody byly ověřeny na testovacích datech a výsledky porovnány. Nejrychleji se přibližné hodnoty konstant podařilo získat gradientní metodou.
Klíčová slova: Genetické programování, Symbolická regrese, Vyčíslování konstant, Gradientní metody.

Abstract: To use the genetic programming for the symbolic regression, the role of constant evaluation is important. The article compares some of commonly used methods. Graphs are included. The Exponentionated gradient descent method showed to be the fastest.
Keywords: Genetic programming, Symbolic regression, Constant evaluation, Gradient descent.

1 Úvod

Jednou z metod, jak k datům nalézt zápis modelu vhodnou funkcí (symbolická regrese) je genetické programování. Genetické programování vychází z genetických algoritmů, které jsou typickým představitelem evolučních výpočetních technik. Aby byla řešitelná operace křížení, aplikoval zde prof. Koza [1] zápis funkce stromem. Potomek vznikne ze dvou rodičů tím, že převezme část stromu jednoho rodiče a doplní jej částí stromu druhého rodiče. Pokud se tvoří dva potomci tak, že se u rodičů vzájemně vymění dva podstromy (větve), mluví se o symetrickém křížení, i když na rozdíl od genetických algoritmů zde nemusí být měněné části stromů stejně velké. Výsledný algoritmus generuje přednostně jedince (navrhované funkce) podobné těm, které jsou nejúspěšnější v průběhu evoluce.

Funkce je programem sestavována z takzvaných primitivních funkcí, za které se zde ale považují i nezávisle proměnné a konstanty. Proměnné a konstanty tvoří listy stromu. Funkce mohou mít jeden parametr (sin, ln, x^2), dva parametry (sčítání, odčítání, násobení, dělení) a obecně i více.

Význam a nutnost vyčíslování konstant vychází až z praktického užití genetického programování ve funkci symbolické regrese. Zatímco školní úlohy jako uhodnutí kombinace funkcí nebo hledání goniometrických ekvivalencí vystačí v roli konstant s jedničkou, dvojkou a nejvýš Ludolfova čísla, v praxi je vyčíslování konstant nevyhnutelné. Například jedna za složek hledané funkce je harmonický signál; kromě určení funkce sinus je třeba ještě vyčíslit amplitudu, frekvenci a fázový posun. Nejčastějším řešením tohoto problému je využití jiné evoluční výpočetní techniky, která je uzpůsobena přímo pro vyhledání optimálních hodnot koeficientů. Většinou však nejde o přímou integraci, ale pomocná metoda vytváří samostatný modul nebo je jinak od hlavní evoluce oddělena.

Tento článek porovnává některé z možných výpočetních metod pro určování hodnot konstant.

2 Data

Aby bylo možné porovnat výsledky jednotlivých metod, byla navržena funkce, pro kterou byly konstanty vyhledávány. Následující text pracuje s funkcí (1)

$$f(x, y) = k_1 \sin(k_2 x + k_3 y) \tag{1}$$

Konstanty byly nastaveny na hodnoty 3,71, 1,73 a poslední má hodnotu 1. Původní záměr byl vyhodnocování jen dvou konstant, ale zde jsou všechny porovnávané metody velice rychlé a spolehlivé.

Data byla vyčíslena ve 350 bodech, pro hodnoty x a y náhodně generované v rozsahu (0; 5). Aby se řešení přiblížilo reálnému problému, byl připočten šum s náhodným rozdělením, daným směrodatnou odchylkou 0,02.

3 Účelová funkce

Účelová funkce je definována jako míra přiblížení navrhovaného průběhu zadaným datům. V tomto případě byl použit nejběžnější [2] součet čtverců odchylek

$$F_j = \sum_{i=1}^{n} \left(\mathrm{G}(x_i, y_i) - f_j(x_i, y_i)\right)^2 = \sum_{i=1}^{n} \left(z_i - f_j(x_i, y_i)\right)^2 \tag{2}$$

Protože se jedná o chybu, má program její hodnotu minimalizovat. Dalšími běžnými funkcemi jsou například součet absolutních hodnot odchylek, nebo tvarové podobnosti. Protože funkce obsahuje šum, nelze dosáhnout nulové hodnoty účelové funkce. Dosazením výše uvedených hodnot, použitých při generování, bylo zjištěno minimum 0,28.

4 Porovnávané metody a výsledky

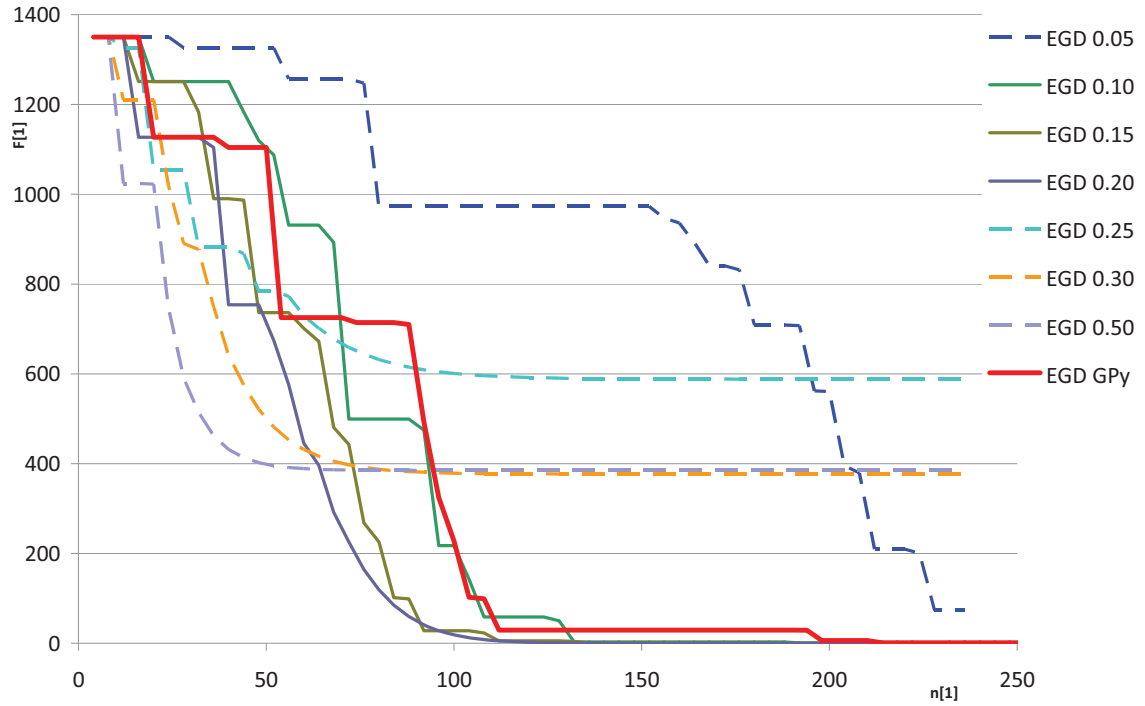
U jednotlivých metod jsou porovnávány současně přesnost a rychlost a výsledky jsou vynášeny do grafů. Jako míra přesnosti je vynášena hodnota účelové funkce. Bez vyčíslení koeficientů (program vygeneruje funkci se všemi koeficienty rovnými jedné) by hodnota účelové funkce při dosazení správné funkce (tj. bez šumu) byla přibližně 1349,8, takže dosažení hodnot blízkých jedné na grafu nelze rozlišit od nuly. Současně je takové řešení možné považovat za dostatečné, s ohledem na účel, pro který je symbolická regrese zpravidla použita.

Vodorovnou osu nemůže reprezentovat čas, protože konkrétní doba provádění záleží na kvalitě implementace a tedy i práci programátora. Vychází se proto z metodiky navržené [3], kdy se za měřítko rychlosti výpočtu považuje počet nutných vyčíslení účelové funkce. Vyčíslení účelové funkce je také zpravidla časově nejnáročnější částí všech algoritmů.

Protože většina metod při opakovaném startu dává rozdílné výsledky, byla každá metoda s každými koeficienty zopakována pětkrát a v grafu je zobrazen ten z průběhů, jehož dosažená výsledná hodnota účelové funkce je střední (medián).

4.1 Gradientní metody

Gradientní metody jsou velmi oblíbeným řešením. Typickou gradientní metodou je i Back Propagation, používaná u neuronových sítí. Gradientní metody pro předem navržené funkce (neuronové jednotky vyšších řádů) používá například [4]. Následující průběhy byly vygenerovány za použití exponencionované gradientní metody [5]. V programu je použit klesající krok algoritmu, což má umožnit nalézt přesnější řešení. Nevýhodou je, že pokud klesá příliš rychle, pak nemusí být celkový počet kroků dostatečný, aby bylo správné řešení opravdu dosaženo. Jednotlivá nastavení reprezentují různé hodnoty koeficientů učení, které jsou ovšem v daném případě přeneseny na klesající krok multiplikativní změny konstant ve směru gradientu pomocí funkce e^x. Program je nastaven tak, že v generované geometrické řadě má třetí krok právě hodnotu konstanty e. Podrobněji viz [6].

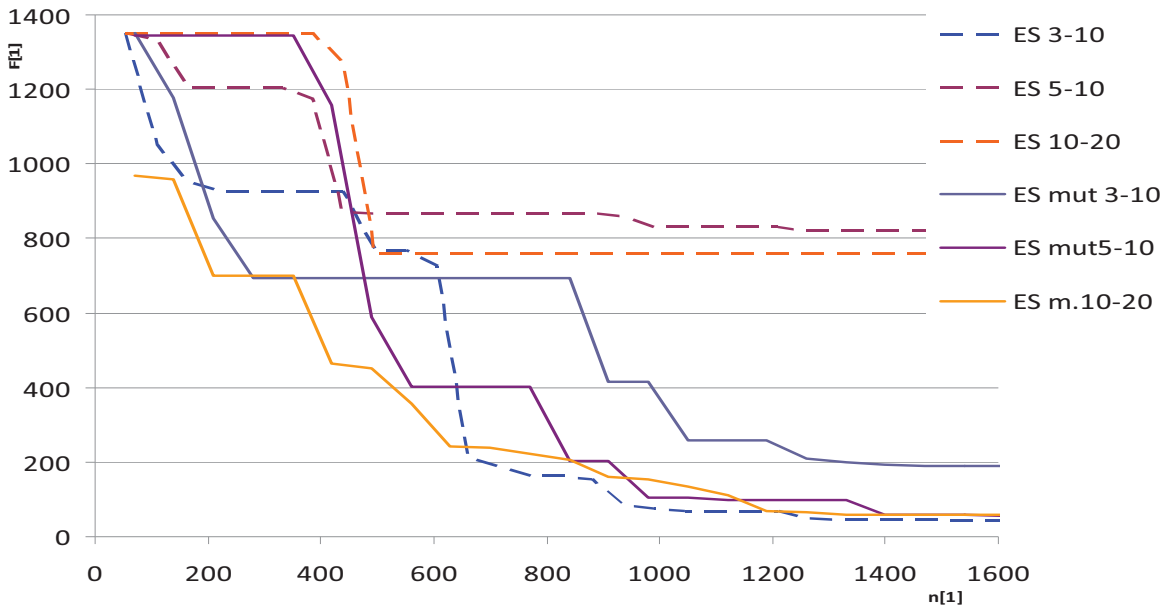


Obr. 1.: Exponencionovaná gradientní metoda. Hodnoty koeficientu učení od 0,10 do 0,20 dávají nejlepší výsledky. Speciální případ je označen GPy, kdy je koeficient učení 0,30, ale metoda je spouštěna vícekrát.

4.2 Evoluční strategie

Evoluční strategie lze podle [7] popsat pravidlem ES(μ/ρ,λ) nebo ES(μ/ρ+λ), kde μ znamená počet jedinců v základní populaci, ρ počet „rodičů“ pro jednotlivá křížení a λ počet generovaných potomků v cyklu. Evoluční strategie pracují s genomem tvořeným vektorem reálných čísel, který popisuje „polohu“ v n-rozměrném prostoru, jsou tedy zvláště vhodné pro řešení hledání kombinace reálných čísel. Pro každý cyklus se vždy λ-krát vybere náhodně ρ rodičů, z těch se vypočte aritmetický průměr, doplní se vhodný šum (realizace mutací) a výsledek se přidá do populace. Na konci cyklu se vybere μ nejlepších pro další cyklus. Ve variantě ES(μ/ρ,λ) se vybírá jen z nově vygenerovaných, ve variantě ES(μ/ρ+λ) i z předchozí generace. Druhá metoda nemůže ztratit již nalezená řešení, první by teoreticky mohla lépe hledat globální extrémy. V průběhu testování se ale nepodařilo metodu ES(μ/ρ,λ) uvést do stavu, kdy by se k řešení blížila, vždy divergovala. Následující grafy uvádějí jen ES(μ/ρ+λ), s různým počtem potomků i se zahrnutím mutací. Evoluční strategie pracují s populací. Počáteční populace je generována v zadaném rozsahu konstant náhodně. Pro všechny metody, kde se generují počáteční populace, byl zvolen interval hodnot (−10; 10), a složky vektorů byly generovány náhodně. Gradientní metody pracují s jediným jedincem a zde použitá metoda vychází z přednastavených konstant, což je v tomto případě vektor jedniček.

U průběhů, kde je v označení mutace, byly výsledné nově generované vektory (jedinci, kombinace konstant) ihned náhodně ponásobením/podělením upraveny v rozsahu ±20%. Tato úmyslná odchylka od průměrné hodnoty vypočtené z vybraných rodičů je základem algoritmu evolučních strategií. Existuje varianta algoritmu, kde dokonce μ=ρ, tedy rodiči jsou všichni jedinci v populaci. Výhodou této varianty je, že v každém cyklu se spočítá průměr jen jednou a všichni jedinci následující generace jsou generováni za využitím mutací. Je zřejmé, že nemůže jít vždy o násobení, protože pokud se jednou průměrná hodnota rodičů v nějaké složce přiblíží nule, evoluce by se zastavila. Pro případ velkých čísel nemůže jít ani výhradně o sčítání. Spolehlivou aplikaci této varianty (konvergující alespoň občas k hledaným hodnotám) se nepodařilo pro danou funkci realizovat.



Obr. 2.: Evoluční strategie, první číslo udává počet rodičů při křížení, druhé velikost populace. Lepší výsledky jsou dosaženy, pokud je nově vygenerovaný jedinec ihned mutován náhodným zmenšením či zvětšením hodnot koeficientů v rozsahu ±20%.

4.3 Genetické programování

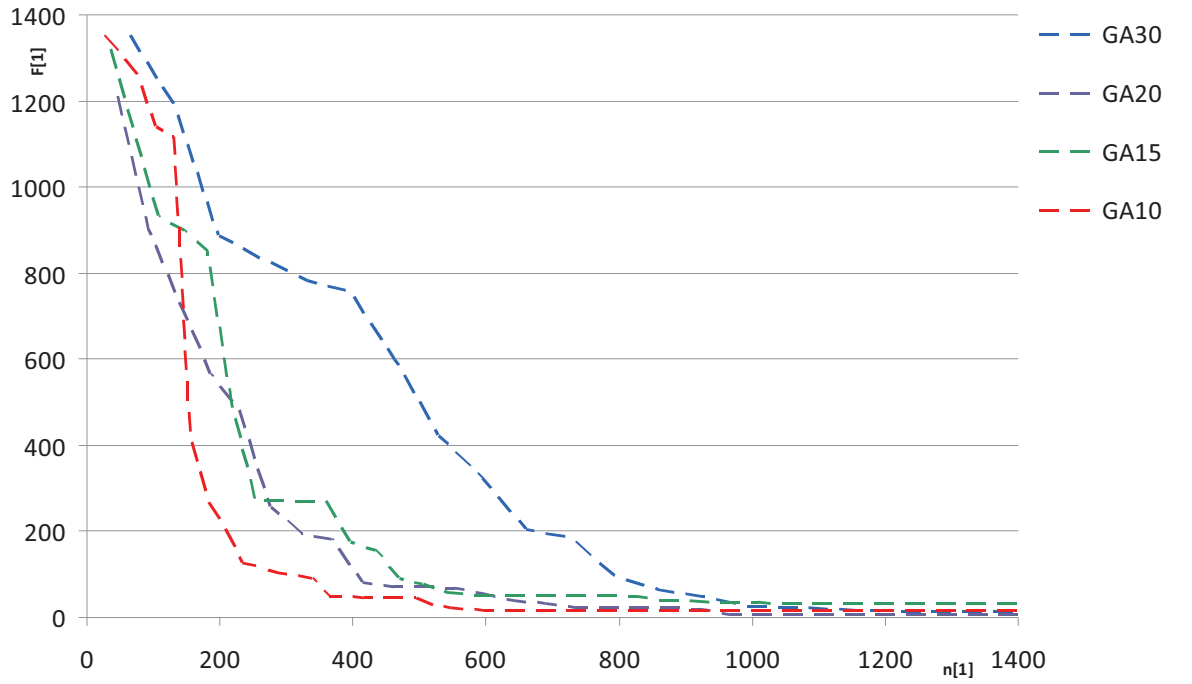
Genetické programování [8] může pracovat s genomem tvořeným přímo reálnými čísly. Pro vytváření potomků je pak používáno tzv. inteligentní křížení [9]. Postup je takový, že dva vybraní jedinci (rodiče) vyznačují v n-rozměrném prostoru úsečku, kde potomek se generuje na této úsečce. Jednotlivé souřadnice jsou dány vzorcem

$$c_i = ka_i + (k-1)b_i \tag{3}$$

a, ***b*** jsou vektory rodičů a ***c*** získaný vektor potomka. Konstanta k je náhodně vygenerované číslo. Metoda konverguje k řešení za předpokladu, že k je generováno nejen okolo 0,5, ale na širším intervalu, nejlépe i mimo původní rodiče, například od –0,3 do 1,3. To ale odpovídá představě křížení, kdy například při křížení dvou různě velkých psů budou štěňata velikostí nejspíše někde mezi, ale mohou být i větší nebo menší než rodiče.

Genetické programování podobně jako evoluční strategie má počáteční populaci a evoluční cykly. Navíc je zde uplatněna tzv. ruletová selekce, která navzdory svému jménu zajišťuje, že nejlepší jedinci jsou vybíráni s větší pravděpodobností, než ti horší. V každém cyklu (generaci) je vytvořen zvolený počet nových jedinců, pak jsou všichni jedinci seřazeni a velikost populace je smazáním nejhorších vrácena na původní velikost a cyklus se opakuje.

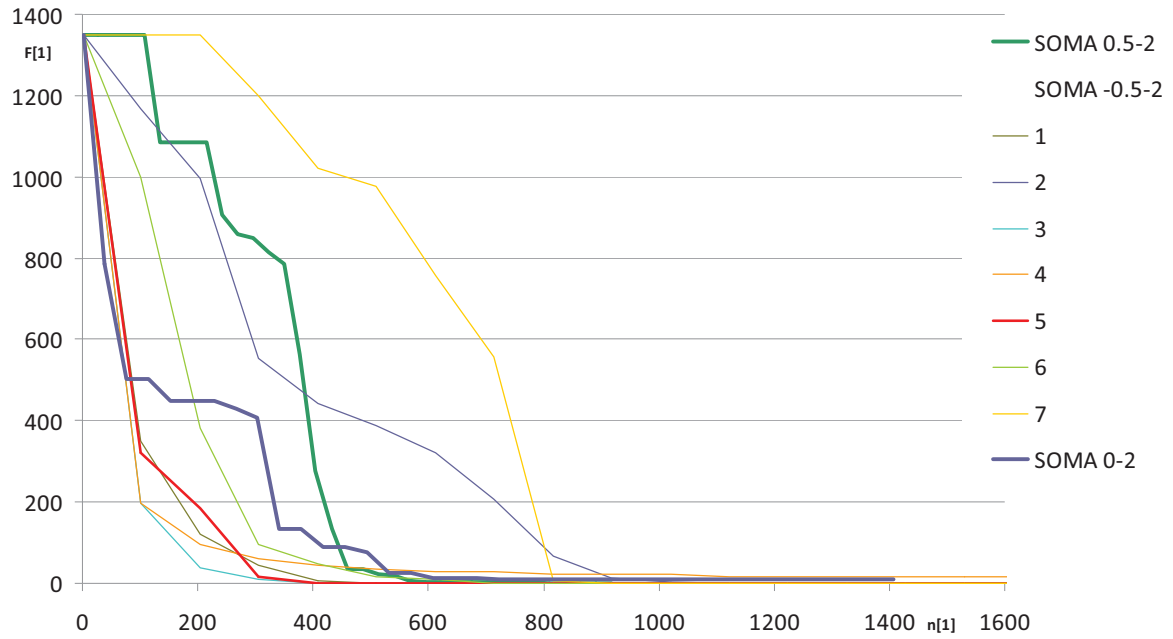
Mutace jsou v genetickém programování řešeny tak, že z populace je vybrán určený počet jedinců (nikdy není vybrán nejlepší jedinec) a u těchto jedinců je náhodně k jedné složce připočtena hodnota, vygenerovaná v rozsahu, v kterém se generovala počáteční populace, nebo je touto hodnotou složka vynásobena (pravděpodobnost obou variant je u vybraného jedince stejná). Jako další mutace může být aplikována náhodná záměna jedné složky vektoru mezi různými jedinci (vzhledem k použitému programu pro genetické programování není tato možnost v „malém“ evolučním cyklu genetických algoritmů aplikována, protože je považována za jednu z mutací ve „velkém“ cyklu genetického programování a aplikace umožňuje volání metody pro upřesňování konstant po každém cyklu genetického programování restartovat).



Obr. 3.: Výsledky získané genetickým programováním. 60 cyklů (nejsou zobrazeny celé), 5 mutací na cyklus, velikost populace je součástí názvu křivky, počet potomků je stejný, jako velikost populace. Je pravděpodobné, že pro malou populaci je třeba více cyklů než u větších populací, vodorovná osa vyznačuje počet vyčíslení účelové funkce a ten je menší. Teoreticky by u příliš malé populace mohlo docházet ke ztrátě diverzity a tím k nenalezení řešení; pokud je řešení nalezeno, je pro malé populace nalezeno rychleji.

4.4 Self-Organizing Migrating Algorithm (SOMA)

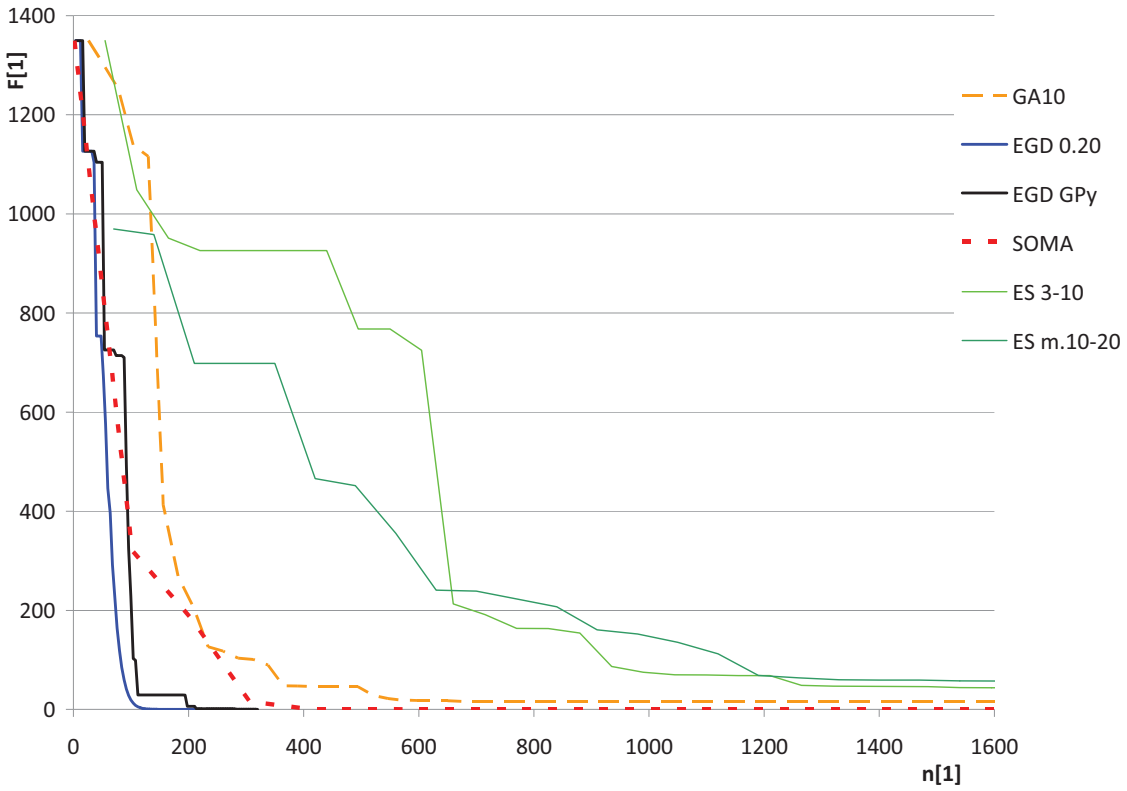
SOMA [10] je migrační algoritmus (patří mezi hejnové algoritmy). Používá počáteční populaci. Ve variantě, jak byl v tomto případě implementován, z ní vybere vždy čtyři jedince. Z nich podle účelové funkce vybere nejlepšího (toho ponechá) a ostatní posune ve směru k tomuto nejlepšímu (označen jako leader, vedoucí). Použije se vzorec pro inteligentní křížení, jen přesah ve směru „za“ nejlepšího je větší. Původní koncepce evokuje, že by se koeficient posunutí na této přímce vždy pohybovat nad délkou vzájemné vzdálenosti, tedy například od jedné do dvou. Následující grafy ukazují průběh evoluce (zde spíše migrace) pro různé hodnoty rozsahu *k*. Sedm průběhů a vyznačený medián (překrývá jeden z průběhů) označuje *k* generované pro každou čtveřici náhodně na intervalu (–0,5; 2,2). Další dvě pak (0; 2,2) a (0,5; 2,2). Celková konvergence je lepší, jsou-li generovány i záporné hodnoty. Nutno podotknout, že v tomto případě nebyly aplikovány mutace.



Obr. 4.: SOMA. Čísla za názvem křivky označují rozsah generování konstanty *k*. Pro rozsah označený (–0,5; 2) je tenkou čarou zobrazeno sedm jednotlivých spuštění a červeně (tlustší čára) je vyznačen medián, pro ostatní je zobrazen jen medián. Rozptyl hodnot je poměrně vysoký.

5 Závěr

V souhrnném grafu (obr. 5) jsou porovnány nejlepší reprezentanti jednotlivých metod. Gradientní metoda je podle předpokladů nejrychlejší. Překvapivě málo výpočtů účelové funkce potřebuje SOMA. Genetické algoritmy jsou podle předpokladů rychlejší, než evoluční strategie. Je to dáno zejména tím, že použitá ruletová selekce zvýhodňuje lepší řešení, aniž by horší zcela vyřazovala, zatímco u evolučních strategií je pravděpodobnost výběru každého z rodičů stejná, pokud zůstanou v dalším cyklu v populaci.



Obr. 5.: Souhrnné porovnání.

Literatura

[1] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA, MIT Press, 1992. ISBN 978-0262111706.
[2] Poli, R., Langdon, W.B., McPhee, N.F., Koza, J. R.: A Field Guide to Genetic Programming. Lulu.com, 2008.
[3] Brandejsky, T.: Small populations in GPA-ES algorithm. In: Mendel 2013, pp. 31-36. ISBN: 978-802144755-4.
[4] Gupta, M.M., I. Bukovsky, N. Homma, M. G. A. Solo, Z.-G. Hou: Fundamentals of Higher Order Neural Networks for Modeling and Simulation. Artificial Higher Order Neural Networks for Modeling and Simulation, pp. 103-133, ed. M. Zhang, IGI Global, 2012.
[5] Kivinen, J., Warmuth, M. K.: Exponentiated gradient versus gradient descent for linear predictors. In: Information and Computation, 132(1):1–63, 1997. doi:10.1006/inco.1996.2612. Available on-line: Elsevier: http://www.sciencedirect.com/science/article/pii/S0890540196926127.
[6] Hlaváč, V.: A program searching for a functional dependence using genetic programming with coefficient adjustment. In: SCSP Prague, 2016.
[7] Hansen, N., D.V. Arnold and A. Auger: Evolution Strategies. In: Janusz Kacprzyk and Witold Pedrycz (Eds.): Handbook of Computational Intelligence, Springer, Chapter 44, pp.871-898 (2015) (pdf: https://www.lri.fr/~hansen/es-overview-2015.pdf).
[8] Mitchell, Melanie: An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press (1996). ISBN 9780585030944
[9] T. Brandejsky: Evolutionary system to model structure and parameters regression. Neural Network World, 22 (2), pp. 181-194, 2012.
[10] Zelinka I.: Analytic Programming by Means of Soma Algorithm. In: Proc. 8th International Conference on Soft Computing Mendel'02, Brno, Czech Republic, 2002, 93-101., ISBN 80-214-2135-5

Selected article from
Tento dokument byl publikován ve sborníku

Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky 2017
New Methods and Practices in the Instrumentation, Automatic Control and Informatics 2017
29. 5. – 30. 5. 2017, Svatý Jan pod Skalou

ISBN 978-80-01-06300-2

Web page of the original document:
http://control.fs.cvut.cz/nmp
http://iat.fs.cvut.cz/nmp/2017.pdf

Obsah čísla/individual articles:
http://iat.fs.cvut.cz/nmp/2017/