Grammar-based genetic programming

Obhajoba diplomové práce

Adam Nohejl

Vedoucí práce: RNDr. František Mráz, CSc.
Katedra software a výuky informatiky, MFF UK Praha

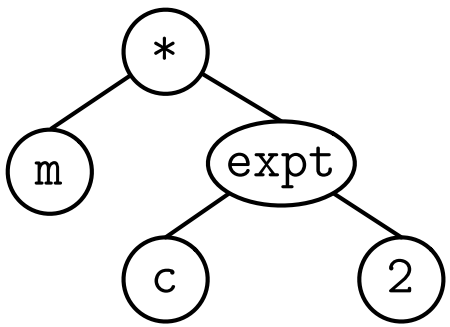
2011

Genetické programování (tradiční verze)

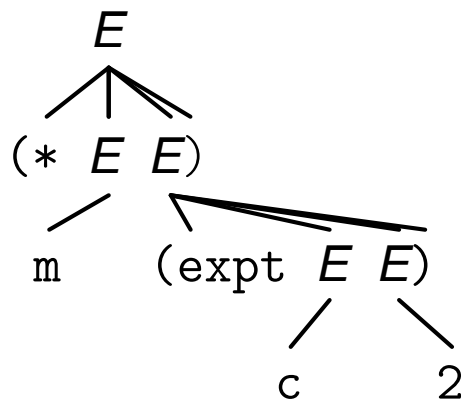
Genetické programování jako druh evolučního algoritmu:

Reprezentace: stromy programů v podmnožině LISPu.

GP strom:



Odpovídající derivační strom:



GP funkce: `*`, `expt` (binární)

GP terminály: 2, `c`, `m`

Odpovídající pravidla gramatiky:

$E \rightarrow$ `(* E E)` $E \rightarrow$ `(expt E E)`

$E \rightarrow 2$ $E \rightarrow$ `c` $E \rightarrow$ `m`

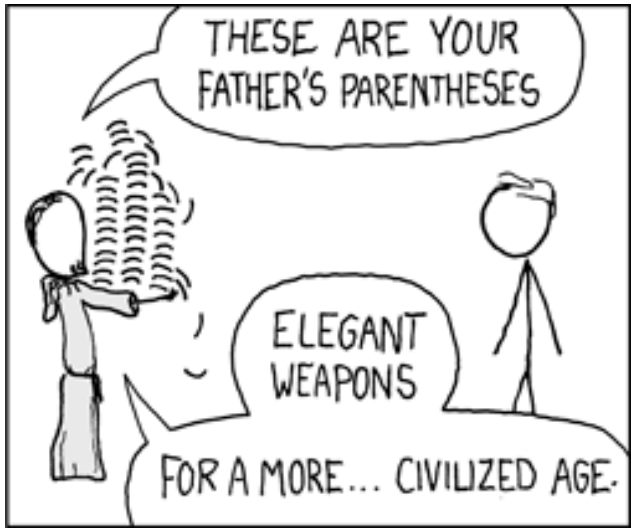
Reprezentovaný výraz: `(* m (expt c 2))`

Operátory: na stromech; respektují množiny funkcí a terminálů.

Zbývá: interpretovat výstup programů, přiřadit fitness.

Genetické programování: problémy

- Jazyk jedinců
  Je vše vhodné programovat v LISPu?
- Typy
  Co když problém zahrnuje více datových typů?
- obecněji: Uzávěr (*closure*)
  Může být každý GP terminál a výstup každé GP funkce libovolným argumentem libovolné GP funkce?



  Dostatečnost (*sufficiency*)
  Umožňují zvolené GP funkce a GP terminály vyřešit problém?
- obecněji: Deklarativní reprezentace znalostí
  Jak zachytit jemnější znalosti o problému a zefektivnit tak jeho řešení?

Genetické programování založené na gramatice

Změny oproti tradičnímu GP:

- místo GP stromů: derivační stromy,
- místo množin funkcí a terminálů: pravidla formální gramatiky.

Přínos: lze popsat jazyk, typový systém i znalosti o problému.

Trade-off: vyjadřovací schopnost $\times$ efektivita operátorů GP.

- CFG-GP (Whigham '95): bezkontextová gramatika (CFG), operace na derivačních stromech.
- LOGENPRO (Wong & Leung '95): logická gramatika, potenciálně Turingovsky úplná, operace na deriv. stromech.
- Grammatical Evolution (Ryan & O'Neill '98): CFG, ale stromy zakódované jako řetězce čísel výběrů v pořadí nejlevější derivace, operace na nich.

Cíle práce

V literatuře chybí srovnání mezi CFG-GP, GE, LOGEPRO.
LOGENPRO a GE byly aplikovány především svými autory.
Operátory GE na číselných řetězcích: efekt na stromy?
Vyplatí se silný formalismus LOGENPRO?

Cíle:

- Popsat, srovnat a (vyjma LOGENPRO) implementovat.
- Porovnat výsledky experimentů v různých aplikacích.

Důraz kladen na

- kvalitní implementaci,
- reprodukovatelnost výsledků,
- ověření proklamovaných výhod metod,
- alespoň empirické zhodnocení efektů operátorů GE.

Logická gramatika: zbytečný luxus

Při pečlivém čtení publikovaných výsledků LOGENPRO se ukazuje, že logické predikáty i unifikace se užívají triviálně $\rightarrow$ zbývá: CFG. Dva experimenty podle Wanga & Leunga (2000) s jejich implementací LOGENPRO a mojí implementací CFG-GP a GE:

- Konečný typový systém lze snadno popsat CFG.
- Generování náhodných konstant pomocí logického predikátu lze nahradit sadou pravidel, navíc je v dané aplikaci zbytečné.
- Celkově: slabší metody CFG-GP a GE dávají podobné výsledky jako LOGENPRO.

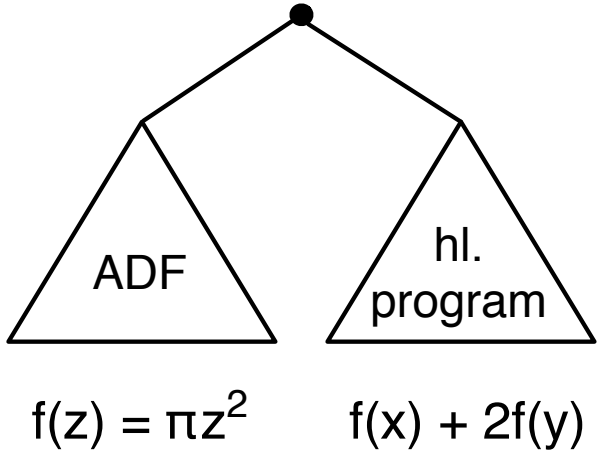
Operátory GE: většinou stejné nebo horší výsledky

Srovnával jsem statistiky výšky a košatosti stromů během generací evoluce u GE a CFG-GP.

- GE má tendenci produkovat nižší stromy, méně košaté, méně různorodých tvarů $\rightarrow$ menší prohledávaný prostor.
- Přitom jsou operátory poměrně destruktivní.

Ve speciálních případech může být chování GE výhodné.
Ve většině experimentů to však vedlo k až o řád horším výsledkům.

Příklad špatného fungování:
automaticky definované funkce („stavební bloky“): podstromy se řídí odlišnými částmi gramatiky.



Obrázek: proměnná z v ADF; funkce f a proměnné x, y v hlavním programu.

GE není nutně časově efektivnější

Nejnáročnější je operace křížení, při které se vyměňují kompatibilní podstromy dvou stromů:

- GE: jednoduchá implementace, navíc asymptoticky: $O(m + n)$, kde m, n jsou počty vrcholů stromů.
- CFG-GP: náročnější implementace, asymptoticky $O(mn)$, ale pokud se implementuje dobře, je praktický rozdíl malý, navíc podstatně náročnější je pak vyhodnocování jedinců.
- LOGENPRO: asymptoticky $O(mn \log m)$ s vysokou multiplikativní konstantou (unifikace) a za předpokladu vyvážených stromů (logaritmická hloubka).

LOGENPRO: velmi náročné pro „pravé“ logické gramatiky, zbytečná režie pro CFG. U GE ale může být celkový běh algoritmu delší než u CFG-GP (uvidíme později).

Rozvrhování zkoušek: pěkné výsledky CFG-GP i GE

Bader-El-Den et al. (2009): článek o zajímavé aplikaci GP s gramatikou (varianta CFG-GP) skoro z reálného světa: rozvrhování zkoušek. Zavedený problém (Carter '96) s mnoha vyzkoušenými heuristikami. Hyperheuristický přístup.
Mé výsledky pro 5 datových sad z 10, kde CFG-GP a GE dopadly zvlášť dobře:

	car91	car92	tre92	uta92	yor83
Bader-El-Den	5.12	4.46	8.62	3.47	40.56
CFG-GP	5.11	4.44	8.78	3.49	40.73
GE	5.15	4.44	8.63	3.51	40.60
CFG-GP*	5.13	4.37	8.63	3.44	40.94
GE*	5.22	4.51	8.79	3.44	41.47
Nejlepší jiná	Fuzzy	Fuzzy	Fuzzy	TM	Fuzzy
	5.20	4.52	8.67	3.04	40.66

*: mnou zvolená gramatika. Nejlepší jiná: z konstruktivních heuristik, podle Bader-el-Den et al. (2009). 1. a 2. nejlepší výsledek. TM: Tabu-Multi-stage.

Shrnutí výsledků

- Efektivní implementace GE a CFG-GP: o řád rychlejší než jiné. Rozsáhlé srovnání těchto metod a LOGENPRO.
- Ve vybraných úlohách (používaných i autory LOGENPRO) se neprokázala výhodnost logických gramatik.
- GE dosahovala obvykle horších výsledků: rozbor příčin.
- V reálné aplikaci rozvrhování zkoušek: podobné výsledky jako Bader-El-Den et al. (2009), s některými sadami dat i lepší (CFG-GP). Ukázal jsem, že lze zvolit výhodnější gramatiku.

Výsledky naznačují, že přínosnější než sofistikované metody GP založeného na gramatikách by mohly být metody generování (evoluce) gramatik vhodných pro úlohu.

Reference

- Mohamed Bahy Bader-El-Den, Riccardo Poli, and Shaheen Fatima. Evolving timetabling heuristics using a grammar-based genetic programming hyperheuristic framework. *Memetic Computing*, 1(3):205–219, 2009.
- Michael W. Carter, Gilbert Laporte, and Sau Yan Lee. Examination timetabling: Algorithmic strategies and applications. *J Oper Res Soc*, 47(3):373–383, 03 1996.
- Michael O'Neill and Conor Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Springer, 2003.
- Peter Whigham. Inductive bias and genetic programming. In *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA*, pages 461–466, 1995.
- Man Leung Wong and Kwong Sak Leung. *Data Mining Using Grammar-Based Genetic Programming and Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.