Automatic Generation and Exploitation of Related Problems in Genetic Programming

Krzysztof Krawiec and Bartosz Wieloch

Institute of Computing Science, Poznan University of Technology, Poland

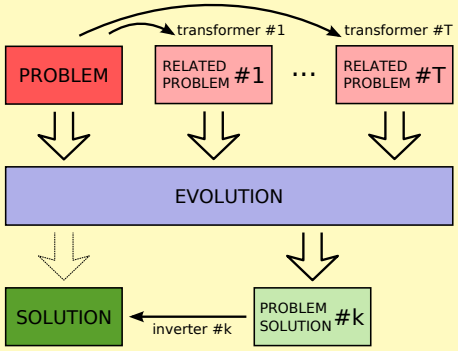
Abstract

We propose an evolutionary framework that uses the set of instructions provided with a genetic programming (GP) problem to automatically build a repertoire of related problems and subsequently employs them to improve the performance of search. The novel idea is to use the synthesized related problems to simultaneously exert multiple selection pressures on the evolving population(s). For that framework, we design two different methods.When applied to six symbolic regression problems of different difficulty, both methods perform better than the standard GP, though sometimes fail to prove superior to a certain control setup.

Idea

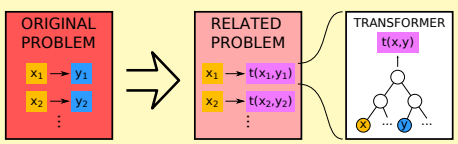
The main idea is to automatically or semi-automatically generate related problems and use them to speed up the process of solving the original problem.

1. Transform in some specific way the original problem to a set of related ones.
2. Run evolution on all problems in parallel, allowing some exchange of genetic material between them.
3. If a solution to one of the related problems is found, use inverse transformation to get solution to the original problem.



Generating related problems

- Input (original) problem is given as a set of *fitness cases* $C = \{(x_i, y_i)\}$ which implicitly defines the fitness function.
- A related problem is defined by a set of fitness cases $C_t = \{(x_i, t(x_i, y_i))\}$ calculated on the pairs from original set C.
- A transformer t is an expression built from the same instructions as the evolved solutions with an additional terminal y returning the desired original value for a given fitness case.
- A transformer has to be *invertible*.



Funding

This work was supported in part by Ministry of Science and Higher Education grant # N N519 3505 33.

Niching Algorithm (NA)

1. For each population P_j, evaluate every individual $s \in P_j$ on each problem C_i. Let $f_i(s)$ denote the fitness of an individual s evaluated on problem C_i (C_i is either the original problem or any derived one).
2. For each problem C_i:
   (a) Create a list A_i of alien individuals from all remaining populations P_j ($j \neq i$), i.e. $A_i = \bigcup_{j \neq i} P_j$
   (b) Independently sort P_i and A_i according to f_i
   (c) Replace the worst $\alpha|P_i|$ individuals from P_i with the best individuals from A_i, $\alpha \in [0, 1]$
   (d) Assign to every individual s from the new population P_i the appropriate fitness $f_i(s)$

Fitness Ranking algorithm (FR)

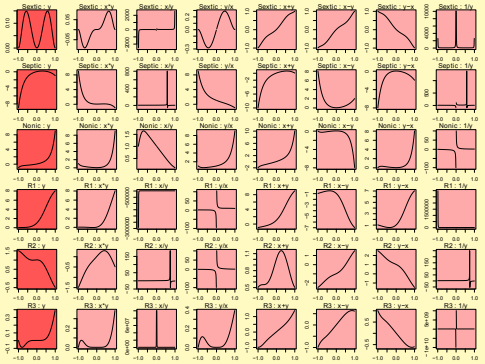
1. Let P denote the current population and $P' \leftarrow \emptyset$ the working set.
2. Evaluate every individual on each problem C_i. Let $f_i(s)$ denote the fitness of an individual s evaluated on problem C_i (C_i is either the original problem or any derived one).
3. For each problem C_i create a ranking R_i of all individuals sorted by $f_i(s)$, i.e., an ordered list of pairs $(individual, rank)$
4. For $i = 0, \ldots, |T|$ do:
   (a) Choose the best individual s from ranking R_i, replace its fitness with its rank in R_i, and add it to P': $P' \leftarrow P' \cup \{s\}$
   (b) Remove s from all rankings (so that all rankings have always the same number of individuals in each step)
5. If there are still individuals in the rankings then go back to Step 4.
6. Set $P \leftarrow P'$

Experiments

Test problems: six univariate symbolic regression benchmarks (three polynomials and three rational functions):

- *Sextic*: $x^6 - 2x^4 + x^2$
- *Septic*: $x^7 - 2x^6 + x^5 - x^4 + x^3 - 2x^2 + x$
- *Nonic*: $x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$
- *R1*: $(x + 1)^3/(x^2 - x + 1)$
- *R2*: $(x^5 - 3x^3 + 1)/(x^2 + 1)$
- *R3*: $(x^6 + x^5)/(x^4 + x^3 + x^2 + x + 1)$

Set of used invertible transformers
$T = \{x * y, x/y, y/x, x + y, x - y, y - x, 1/y\}$



Parameter	Value
fitness cases	20, $x \in [-1, 1)$
functions	only $+, -, *, /$
individuals	1024
crossover	0.9
mutation	0.1
generations	max 300

Conducted experiments:

GP1 standard GP, all 1024 individuals evolve in a single population
GP8 standard GP, but individuals evolved in 8 populations of size 128
NA niching algorithm with different α values
FR fitness ranking algorithm

Results

Averaged success ratio from 300 runs:

Frequencies of ideals found using particular transformers, averaged over all NA experiments and all values of α:

Problem	Transformer							
	y	$x * y$	x/y	y/x	$x + y$	$x - y$	$y - x$	$1/y$
Sextic	36.7	19.9	2.1	34.2	4.6	5.8	3.6	2.1
Septic	25.4	9.9	1.4	21.3	14.0	14.6	12.1	0.7
Nonic	13.9	9.3	1.2	16.3	8.1	7.6	40.6	1.7
R1	13.0	9.0	32.1	4.9	1.5	2.5	2.1	33.7
R2	29.5	13.4	1.7	17.7	18.6	5.6	10.3	2.3
R3	11.9	8.6	37.8	16.6	3.1	2.5	2.7	16.2

Conclusions

- NA performs best for difficult rational problems (R1, R2, and R3), especially for high α.
- FR generally performs worse than NA but is parameter-free.
- GP8 is the best method for easiest polynomial problems (Sextic, Septic, and Nonic)
- Single-population approach (GP1) is almost always the worst.

General conclusion: solving in parallel the original problem and the automatically generated related problems, with exchange of some genetic material between them, typically increases the success rate.