Locally Geometric Semantic Crossover

Krzysztof Krawiec
krawiec@cs.put.poznan.pl

Tomasz Pawlak
tomasz.pawlak@cs.put.poznan.pl

Institute of Computing Science, Poznan University of Technology
Piotrowo 2, 60965 Poznań, Poland

ABSTRACT

We propose Locally Geometric Crossover (LGX) for genetic programming. For a pair of homologous loci in the parent solutions, LGX finds a semantically intermediate procedure from a previously prepared library, and uses it as replacement code. The experiments involving six symbolic regression problems show significant increase in search performance when compared to standard subtree-swapping crossover and other control methods. This suggests that semantically geometric manipulations on subprograms propagate to entire programs and improve their fitness.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

Keywords

genetic programming, program semantics, semantic crossover

1. MOTIVATION AND THE METHOD

Standard search operators used in genetic programming (GP) are blind to program semantic. For them, programs are purely symbolic structures, where opcodes associated with particular instructions have no particular *meaning*. Given that knowledge of semantics is one of factors that makes human programming so effective, equipping GP algorithms in some semantic-aware extensions could have positive impact on their performance.

We devise a method that makes GP alert to certain semantic aspects of programs. For this, we assume that fitness function is a *metric* $||\,||$ that captures the divergence (error) between program output and some known desired output, which is typically the case in GP. Consequently, the fitness landscape is a *convex surface* spanned over the space of vectors that hold program outputs.

Convexity allows designing recombination operators that are likely to yield offspring of good quality [6]. We exploit this property *locally*, on the level of program parts, rather than entire programs. The method operates in two phases. First, a library L of short *procedures* of height at most h is generated. Next, we calculate the *semantics* $s(p)$ of every procedure $p \in L$, meant here as the vector of outcomes it produces for the fitness cases. For any two procedures p_1, p_2 that have the same semantics ($||s(p_1) - s(p_2)|| = 0$) we discard the longer one to avoid redundancy.

In the second phase, the library is used by the proposed *locally geometric crossover* (LGX) during a GP run. Given two parent programs p_1 and p_2, LGX first identifies the structurally *common region* [7], i.e., the set of tree node locations that occur in both parents. Then LGX appoints a single random locus in the common region as a crossover point. To reduce bloat, this follows the same rules as in standard GP [2]: with probability 0.1, the locus is a leaf in the common region, otherwise an internal node.

Let p_1' and p_2' denote the subtrees rooted in the drawn locus in p_1 and p_2, respectively. We calculate their semantics, $s(p_1')$ and $s(p_2')$, and determine the midpoint between them in the semantic space: $s_m = (s(p_1') + s(p_2'))/2$. This point represents the semantics of a hypothetical procedure $p : s_m = s(p)$ which, if inserted into parents at the appointed locus, would make the resulting offspring semantically intermediate at that locus. However, finding p in general requires solving an inverse problem $p = s^{-1}(s_m)$, which is a separate program induction problem itself. Also, such p may happen to not exist, or be very long.

Thus, instead of looking for a procedure whose semantics is exactly s_m, we find in the library L the procedure that is semantically *most similar* to s_m: $p = \arg\min_{p' \in L} ||s(p') - s_m||$. The procedure p replaces then the subtrees p_1' and p_2' in the parent solutions, and the modified trees become the offspring. To efficiently search for the semantically most similar procedure in the library, we employ *spatial index*, a data structure designed for geographic databases. From variety of spatial indexes, we have chosen *R-tree* [1], due to its ability to work under arbitrary chosen norm and dimensionality.

2. THE EXPERIMENT

We verify whether the semantic properties of LGX influence search efficiency by solving univariate symbolic regression problems shown in Table 2, taken from [4], using instruction set $\{+, -, \times, /, x\}$. Semantics is defined as a vector of values returned by a program for 20 fitness cases distributed equidistantly in the interval $[-1, 1]$. We consider two libraries, for the maximum procedure (tree) height $h = 3$ (81 procedures, in this 38 semantically distinct, so $|L| = 38$), and $h = 4$ (21385 procedures, 1697 semantically distinct).

We confront LGX with canonic GP [2], using conventional tree-swapping crossover that uses the same probability distribution as LGX for node selection (0.1 for leaves and 0.9

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

Table 1: Absolute error with 0.95 conf. interval committed by the best-of-run individuals (avg. of 150 runs).

	Sextic		*Septic*		*Nonic*		*R1*		*R2*		*R3*	
GP	0.002	±0.001	0.159	±0.040	0.122	±0.041	0.441	±0.102	0.231	±0.040	0.184	±0.026
RX_3	0.002	±0.001	0.130	±0.039	0.128	±0.040	0.175	±0.039	0.130	±0.028	0.164	±0.027
LGX_3	0.003	±0.001	0.109	±0.034	0.114	±0.041	0.170	±0.033	0.086	±0.020	0.179	±0.024
RX_4	0.005	±0.002	0.102	±0.024	0.130	±0.035	0.140	±0.035	0.101	±0.019	0.076	±0.014
LGX_4	0.001	±0.001	0.044	±0.011	0.043	±0.009	0.061	±0.014	0.041	±0.013	0.028	±0.007

for internal nodes). Like all other operators considered here, it never replaces the root node. The other control setup is *random crossover* (RX), intended to verify whether the observed performance is an effect of the geometric character of LGX or potentially some other factor. RX operates like LGX, but chooses a procedure from L at random. Thus, RX is similar to LGX in terms of mode of operation, but it is completely blind to the structure of semantic space. There are thus 5 methods in total: GP, LGX_3, LGX_4, RX_3 and RX_4, where indices denote h.

Each run starts from a different initial population of 1024 individuals and lasts for 250 generations. Fitness is the absolute error of the individual's output w.r.t. the desired output, summed for the 20 fitness cases. We use tournament selection of size 7, crossover with likelihood 0.9, and reproduction with likelihood 0.1. There is no mutation involved. Other parameters are set to ECJ's defaults [5].

Table 1 shows the average best-of-run fitness. LGX_4 achieves the best performance for all benchmarks. LGX_3 finds noticeably poorer solutions, probably due to the almost two orders of magnitude smaller library. Consequently, the procedures it inserts into offspring are semantically less diversified, which causes exploration to be less intense.

Surprisingly, RXs turn out to be sometimes superior to GP. We have two hypotheses to explain this phenomenon. Firstly, RX can compensate for the lack of mutation by constantly supplying population with diverse genetic material from the library. GP, devoid of mutation, is forced to use only the genetic material from population. However, this is unlikely to be the reason for $h = 3$, when the library contains mere 38 procedures. The other explanation is that GP, by acting on deeper tree nodes, introduces semantically neutral changes more frequently than LGX and RX that operate in the common region, and thus in the shallower tree parts.

Most importantly however, RXs almost never substantially surpass LGXs. This suggests that introducing 'medial' tendency in crossover makes the search process converge faster towards good solutions. The effects of semantic-aware changes introduced into inner tree nodes propagate to the root, and improve the fitness of offspring more than for the other methods. This confirms the conclusion of our former study on discrete problems using linear programs [3].

Table 2: Test problems.

Problem	Definition (formula)
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	$(x^6 + x^5)/(x^4 + x^3 + x^2 + x + 1)$

On the other hand, the fitness gap between LGX and the control setups, albeit statistically significant for most problems, is usually small. By operating on subtrees that can be located deeply in parent trees, LGX cannot improve fitness every time it is applied.

3. CONCLUSION

Crossover that operates geometrically on the level of semantics *of homologous subtrees* can improve the efficiency of GP search. We hypothesize that, over time, LGX helps forming a common semantic blueprint in the population, with subprograms located in particular loci specializing in solving certain subproblems of the original problem. This suggests capability to discover semantic modules in the structure of the problem, which in turn could provide possibility of problem decomposition.

Acknowledgment. Work supported by grants no. DEC-2011/01/B/ST6/07318 and 91-528/12-DS.

4. REFERENCES

[1] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Conf.*, page 47, Boston, MA, June 1984. Reprinted in M. Stonebraker, Readings in Database Sys., Morgan Kaufmann, San Mateo, CA, 1988.

[2] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[3] K. Krawiec. Medial crossovers for genetic programming. In A. Moraglio, S. Silva, K. Krawiec, P. Machado, and C. Cotta, editors, *Proceedings of the 15th European Conference on Genetic Programming, EuroGP 2012*, volume 7244 of *LNCS*, pages 62–73, Malaga, Spain, 11-13 April 2012. Springer Verlag.

[4] K. Krawiec and B. Wieloch. Automatic generation and exploitation of related problems in genetic programming. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, Barcelona, Spain, 18-23 July 2010. IEEE Press.

[5] S. Luke. *The ECJ Owner's Manual – A User Manual for the ECJ Evolutionary Computation Library*, zeroth edition, online version 0.2 edition, Oct. 2010.

[6] A. Moraglio, K. Krawiec, and C. Johnson. Geometric semantic genetic programming. In C. Igel, P. K. Lehre, and C. Witt, editors, *The 5th workshop on Theory of Randomized Search Heuristics, ThRaSH'2011*, Copenhagen, Denmark, July 8-9 2011.

[7] R. Poli and W. B. Langdon. Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation*, 6(3):231–252, 1998.