Solving the Lawn Mower problem with Kaizen Programming and λ-Linear Genetic Programming for Module Acquisition

Léo Françoso dal Piccol Sotto
Institute of Science and Technology (ICT)
Federal University of São Paulo (UNIFESP)
São José dos Campos, SP, Brazil
leo.sotto@unifesp.br

Vinícius Veloso de Melo
Institute of Science and Technology (ICT)
Federal University of São Paulo (UNIFESP)
São José dos Campos, SP, Brazil
vinicius.melo@unifesp.br

ABSTRACT

In this work, we have tested a new approach for evolving modular programs: Kaizen Programming (KP) with λ-Linear Genetic Programming (λ-LGP) and a heuristic search procedure to solve the well-known Lawn Mower problem. KP is a novel hybrid approach that tries to efficiently combine partial solutions to generate a high-quality complete solution. Being a hybrid, KP may use different types of methods to generate partial solutions, assess their importance to the complete solution, and solve the complete problem. Experiments on the Lawn Mower problem show that the proposed method is effective in finding the expected solution. It is a new alternative for evolving modular programs, but further investigations are necessary to improve its performance.

CCS Concepts

•Computing methodologies → Artificial intelligence; Genetic programming; •Software and its engineering → Automatic programming;

Keywords

Kaizen Programming, Collaborative Problem Solving, Genetic Programming, Modularity, Lawnmower Problem

1. INTRODUCTION

The Lawn Mower problem was proposed by Koza in order to evaluate Automatically Defined Functions (ADFs), which was proposed to deal with modularity in Genetic Programming (GP) [4]. This benchmark problem has a virtual lawn mower which must be programmed to "virtually mow a grass lawn" consisting of a toroidal grid with $n \times m$ squares. The terminal set has the functions *left*, *mow*, and R_{v8}, while the function set contains the functions *V8A*, *FROG*, and *PROGN*, as shown in [7].

GECCO '16 Companion, July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2909007

The Lawn Mower problem has a modular nature; the repetition of some blocks of code is a pattern to the mower to cut all the grass in the rectangular area. For such reason, most works that solve this problem try some form of module acquisition and reuse, as can be seen in [7, 6, 8, 9].

Here, we propose a combination of the λ-Linear Genetic Programming (λ-LGP) [5] and Kaizen Programming (KP) [2]. λ-LGP is an LGP [1] that applies λ operations (mutations) on each individual at each generation, replacing them in the population according to certain criteria; thus, promoting a kind of "intensive" search. KP is a hybrid approach that uses specialist procedures to generate partial solutions, and combines them to construct a complete solution to a given problem, such as regression [3]. The partial solutions that contribute the most to the complete solution are chosen to the next iteration, resulting in a continuous improvement cycle. In this work, we use λ-LGP to provide the specialists to KP.

With the proposed approach, we introduce modularity to improve the performance of LGP for solving the Lawn Mower problem. It works as follows: 1) A population (called *standard*) of size N is initialized; each individual - (partial solution) is a *module*, a block of code; 2) At each generation, λ mutations are applied to each individual, resulting in an intermediate population of size N*λ; 3) A trial solution is *greedly* constructed by testing which individual of this population better improves the performance when concatenated to the current Trial Solution (initially empty), until reaching a size L of individuals; 4) If the new Trial Solution is better than the current *standard*, it substitutes it, and N individuals are chosen to pass to the next cycle; 5) Half of the individuals are the ones that contribute the most when concatenated to the Trial Solution (calculated during its construction), and half are the best fitted individuals - ignoring the combination.

2. EXPERIMENTS

We first implemented the standard LGP with effective macro and micro-mutations, with a 75% rate for the former and 25% for the latter. We implemented λ-LGP and KP using LGP, varying the *standard's* size (maximum number of individuals that can compose the complete solution) as 3, 5, and 7. An individual's size starts with 20 and can grow up to 200 instructions. For KP, as one wishes to evolve smaller modules, the size varied between 10 and 15. We tested population sizes of 10, 50, and 100. We present results

for population=10 as it showed the best results. We used $\lambda = 10$ and maximum number of generations= 200. We tested the techniques in lawns of four sizes: 8x8, 12x12, 14x14, 16x16. The maximum fitness for each map is its total area, while the maximum number of steps allowed obeys the relation $MaxS = 1.5625 * area$, as used in [7]. Table 1 shows the results obtained by regular LGP and KP λ-LGP over 100 executions.

Table 1: Mean fitness, success rate, minimum effort (I/1000), and generation in which the minimum effort was obtained using LGP and KP λ-LGP (multiplied by $\lambda * N$) with different solution sizes (N) in different map sizes.

Map/*MaxS*	Method/N	*Fitness*	Success	I/1000	Gen
8x8/100	LGP	64	1	7.4	36
	KP λ-LGP/3	64	1	15.3	16
	KP λ-LGP/5	64	1	25	24
	KP λ-LGP/7	63.99	0.99	39.9	18
12x12/225	LGP	143.99	0.99	18.8	93
	KP λ-LGP/3	143.99	0.99	27.9	30
	KP λ-LGP/5	143.83	0.97	57.5	22
	KP λ-LGP/7	143.99	0.99	63	44
14x14/306	LGP	195.98	0.99	14.6	72
	KP λ-LGP/3	195.65	0.95	40.5	26
	KP λ-LGP/5	195.90	0.99	58.5	38
	KP λ-LGP/7	195.47	0.93	100.8	23
16x16/400	LGP	255.96	0.99	17.8	88
	KP λ-LGP/3	255.98	0.99	27.9	30
	KP λ-LGP/5	256	1	62.5	24
	KP λ-LGP/7	255.71	0.98	65.8	46

As Table 1 suggests, the two techniques were successful in finding the optimal solution, achieving the maximum mean fitness or very near to it (see the success rate). LGP alone was able to solve the problem, but using KP to combine and evolve individual modules, one intends to solve the problem faster. As shown in the Generations (Gen) column, KP finds the solutions earlier - sometimes 3 times earlier when using a *standard* of size 3. Nevertheless, the computational efforts were greater than the obtained by LGP. The reason is that KP tests the Trial Solution after each individual is concatenated. As the maximum size influences this mechanism, bigger values for such parameter deteriorated the performance. However, even though it is not in this report, KP outperformed several related works from the literature.

Figure 1 shows the evolution curves for the best individual (partial solution) in the population and the complete solution. The curves are shown only for the 16x16 map due to the short page limit of this paper, but other maps follow the same pattern. It is clear that the complete solution gets fast to the global optimum, while the best individual improves slowly.

3. CONCLUSIONS

We confirmed that, for the investigated problem, KP demonstrates a good potential of finding high-quality solutions earlier than LGP by combining partial solutions. Further developments, like using smaller values for λ, or modifying the mechanism for constructing the Trial Solution, are to be investigated in order to improve the performance of the method.

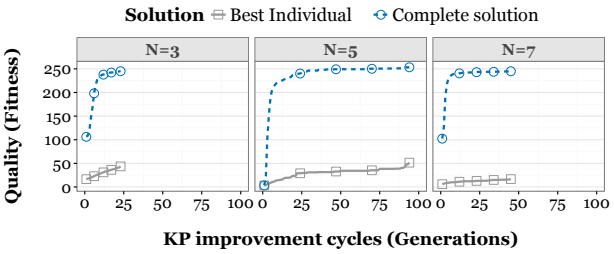

Figure 1: Evolution curves (mean of 100 independent runs).

4. ACKNOWLEDGMENTS

This paper was supported by grants #2013/20606-0 and #2016/04946-4, São Paulo Research Foundation (FAPESP), the Brazilian Government grants #486950/2013-1 (CNPq Universal) and #12180-13-0 CAPES (Science without Borders program).

5. REFERENCES

[1] M. F. Brameier and W. Banzhaf. *Linear Genetic Programming*. Springer, 2007.

[2] V. V. De Melo. Kaizen programming. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 895–902, New York, NY, USA, 2014. ACM.

[3] V. V. de Melo and W. Banzhaf. Predicting high-performance concrete compressive strength using features constructed by kaizen programming. In *2015 Brazilian Conference on Intelligent Systems, BRACIS 2015, Natal, Brazil, November 4-7, 2015*, pages 80–85, 2015.

[4] J. R. Koza. Genetic programming ii: Automatic discovery of reusable subprograms. *Cambridge, MA, USA*, 1994.

[5] L. F. Sotto, V. V. De Melo, and M. P. Basgalupp. An improved λ-linear genetic programming evaluated in solving the santa fe ant trail problem. In *Proceedings of the Symposium on Applied Computing*, April 2016, Forthcomming.

[6] L. Spector and S. Luke. Cultural transmission of information in genetic programming. In *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 209–214, Cambridge, MA, USA, 1996. MIT Press.

[7] L. Spector, B. Martin, K. Harrington, and T. Helmuth. Tag-based modules in genetic programming. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO'11, pages 1419–1426, Dublin, Ireland, 12-16 July 2011. ACM.

[8] J. M. Swafford, E. Hemberg, M. O'Neill, M. Nicolau, and A. Brabazon. A non-destructive grammar modification approach to modularity in grammatical evolution. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO'11, pages 1411–1418, New York, NY, USA, 2011. ACM.

[9] J. Walker and J. Miller. The automatic acquisition, evolution and reuse of modules in cartesian genetic programming. *Evolutionary Computation, IEEE Transactions on*, 12(4):397–417, Aug 2008.