Late-Acceptance Hill-Climbing with a Grammatical Program Representation

James McDermott
College of Business
University College Dublin
Dublin, Ireland D04 V1W8
jmmcd@jmmcd.net

Miguel Nicolau
College of Business
University College Dublin
Dublin, Ireland D04 V1W8
miguel.nicolau@ucd.ie

ABSTRACT

The late-acceptance hill-climbing (LAHC) metaheuristic is a stochastic hill-climbing algorithm with a simple history mechanism, proposed by Burke and Bykov in 2008, which seems to give a remarkable and reliable performance improvement relative to hill-climbing itself. LAHC is here used for the first time for genetic programming problems, with a grammatical encoding. A novel variant of LAHC with an initial random sampling is also proposed. Performance of both is competitive with full population-based search.

CCS CONCEPTS

•**Computing methodologies → Genetic programming;**

KEYWORDS

Genetic programming, grammar, hill-climbing

ACM Reference format:
James McDermott and Miguel Nicolau. 2017. Late-Acceptance Hill-Climbing with a Grammatical Program Representation. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3067695.3075984

```python
def LAHC(L, n, C, init, mutate):
    s = init()                # initial solution
    Cs = C(s)                 # cost of current solution
    best = s                  # best-ever solution
    Cbest = Cs                # cost of best-ever
    f = [Cs] * L              # initial history
    for I in range(n):        # number of iterations
        s_ = mutate(s)        # candidate solution
        Cs_ = C(s_)           # cost of candidate
        if Cs_ < Cbest:
            best = s_         # update best-ever
            Cbest = Cs_
        v = I % L             # v indexes f circularly
        if Cs_ <= f[v] or Cs_ <= Cs:
            s = s_            # accept candidate
            Cs = Cs_          # (otherwise reject)
        f[v] = Cs             # update circular history
    return best, Cbest
```

Figure 1: Working Python code for LAHC.

1 INTRODUCTION

It has been observed that mutation-only GP can perform surprisingly well [2, 5]. An important motivation for mutation-only algorithms is simplicity: there are few parameters to tune. In this paper, we focus on a mutation-only algorithm, *late-acceptance hill-climbing* (LAHC) [1], which seems to provide a remarkable performance improvement relative to basic hill-climbing.

2 SEARCH ALGORITHMS AND ENCODING

The main feature of LAHC is a history of objective function values, denoted f, of length L. At each step, a candidate solution is obtained by mutation from the current solution. If its cost is better than or equal to the current solution, or better than or equal to that which occured L steps previously, the candidate solution is accepted and becomes the current solution; otherwise it is rejected. Either way, the appropriate item in f is updated with the current objective function value. To emphasise the simplicity of LAHC, we provide working Python code (see Figure 1).

LAHC has not previously been used for any GP problem. We will compare LAHC (with a GE encoding) to a population-based search, GE itself [4].

GE uses a population, and LAHC does not. A population has two effects on evolution: it provides a large *initial* sampling, and it changes the nature of *ongoing* search, since it allows parallel search and "information transfer" via crossover. In order to try to disentangle these two effects, we implemented a novel feature in LAHC: an initial sampling, from which the best individual is chosen as the first "current" individual in the LAHC search. (Of course, the initial sampling counts towards the LAHC fitness evaluation budget.) In our runs, LAHC denotes the "vanilla" algorithm, and LAHC-S denotes the variant with initial sampling.

The initial populations for GE were created using Sensible Initialisation [6]. In the case of GE, the initial tree depth was ramped within the range specified in the experimental setup (see Table 1). In the case of LAHC/LAHC-S, all individuals were initialised using the maximum initial tree depth, to prevent the creation of very small initial individuals. The reason for this is that in the mutation-only algorithms, genomes never change in length. Therefore, we initialise individuals to a *large* depth hence large genome length, and allow shorter genomes to occur in effect by not using all the genotypic material during the mapping process.

GECCO '17 Companion, Berlin, Germany
 978-1-4503-4939-0/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3067695.3075984

3 EXPERIMENTS AND RESULTS

We evaluate our systems on six symbolic regression problems (Vladislavleva-4, Pagie-1, Keijzer-6, Dow, Tower, and Housing with dimensions 5, 2, 1, 57, 5, 13) [3]. Parameters are shown in Table 1.

Table 1: Experimental Setup

Parameter	GE	LAHC	LAHC-S
Evaluation budget	25000*	25000*	25000*
Population Size	500*	1	1
Generations	Budget divided by pop. size		
Initialisation	Sensible		Sensible
Initialisation Depths	2–5	5–5	5–5
Initialisation Tails	$0.5 \times l$	$10 \times l$	$10 \times l$
Tournament Size	1%		
Crossover prob.	50%		
Wrapping	Off	Off	Off
Integer mutation prob.	$1.0/l$	$1.0/l$ with min. 1 event	
Elitism	1%		

* For V4, Housing and Dow: budget 50000, pop. size 1000.

Sample results are shown in Fig. 2: others are omitted for brevity. For the Housing, Tower, and Dow datasets, all search methods struggle to compete with a linear regression (*LR*) baseline. Even predicting a constant (*const*) baseline out-performs some setups. The same has been observed (on the Tower problem) with a state-of-the-art symbolic regression method [7].

For most problems, on both training and test performance, the best setups are LAHC and LAHC-S with intermediate values ($L = 50, 100, 500$), and standard GE. Overall, standard GE may have the edge, but it is marginal and somewhat problem-dependent. There is no great difference between LAHC and LAHC-S. Although LAHC-S is a novelty proposed in this paper, this result is positive because it means that the simplest variant of LAHC remains competitive.

Fig. 3 analyses the effect of L using LAHC (results with LAHC-S are similar). $L = 100$ dominates, with $L = 50$ and $L = 500$ runners-up. Thus when L is too small, search becomes stuck at local optima ($L = 1$ reduces to simple hill-climbing); when L is too large, too many disimproving moves are accepted, and there is not enough true "climbing" behaviour. Thus, L gives a practical control of the exploration-exploitation balance.

4 CONCLUSIONS

We have for the first time combined a late-acceptance hill-climbing (LAHC) search algorithm with a grammar-based genetic programming encoding. We have tested this combination against a more typical genetic programming search algorithm, and found that despite its simplicity (about 20 lines of code, and just one tunable parameter), LAHC performs surprisingly well. It is a useful addition to the GP toolbox.

The addition of an initial sampling (LAHC-S) does not greatly change long-term LAHC performance, hence differences between LAHC and GE can instead be attributed to ongoing information transfer via crossover.

Several avenues for future research are opened up by this research, especially using LAHC search with other GP encodings.

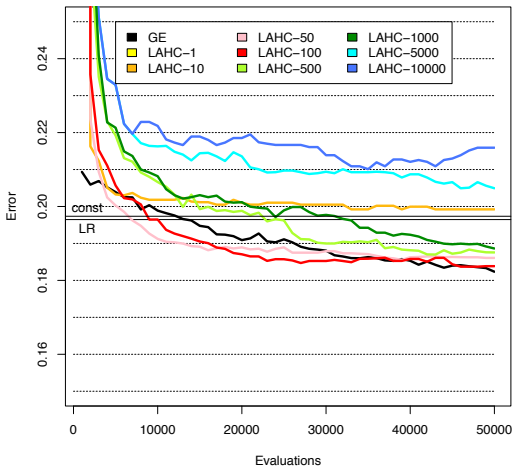

Figure 2: Test performance on Vladislavleva-4. LAHC-1 is off the chart. *Const* and *LR* baselines are shown.

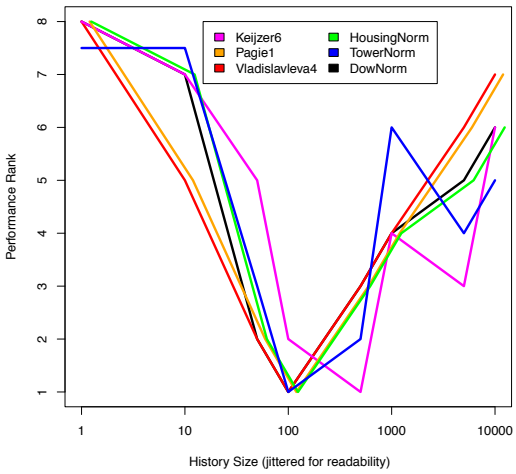

Figure 3: L: ranked test performance with LAHC.

REFERENCES

[1] Edmund K Burke and Yuri Bykov. 2008. A late acceptance strategy in hill-climbing for exam timetabling problems. In *PATAT 2008 Conference, Montreal, Canada.*

[2] Kumar Chellapilla. 1997. Evolutionary programming with tree mutations: Evolving computer programs without crossover. In *Genetic Programming* (July 13–16 1997). Stanford, CA, USA, 431–438.

[3] James McDermott, David R. White, Sean Luke, Luca Manzoni, Mauro Castelli, Leonardo Vanneschi, Wojciech Jaśkowski, Krzysztof Krawiec, Robin Harper, Kenneth De Jong, and Una-May O'Reilly. 2012. Genetic Programming needs better benchmarks. In *Proceedings of GECCO 2012.* ACM, Philadelphia.

[4] Michael O'Neill and Conor Ryan. 2003. *Grammatical Evolution - Evolutionary Automatic Programming in an Arbitrary Language.* Genetic Programming, Vol. 4. Kluwer Academic.

[5] Una-May O'Reilly and Franz Oppacher. 1994. Program Search with a Hierarchical Variable Length Representation: Genetic Programming, Simulated Annealing and Hill Climbing. In *PPSN III (Lecture Notes in Computer Science)*, Yuval Davidor et al. (Ed.). Springer-Verlag, Jerusalem, 397–406.

[6] Conor Ryan and Atif Azad. 2003. Sensible Initialisation in Grammatical Evolution. In *GECCO*, Erick Cantú-Paz and others (Eds.). AAAI.

[7] E.J. Vladislavleva, G.F. Smits, and D. Den Hertog. 2009. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *Trans. Evol. Comp.* 13, 2 (2009), 333–349.