Predict the Success or Failure of an Evolutionary Algorithm Run

Gopinath Chennupati
BDS Group
CSIS Department
University of Limerick, Ireland
gopinath.chennupati@ul.ie

Conor Ryan
BDS Group
CSIS Department
University of Limerick, Ireland
conor.ryan@ul.ie

R. Muhammad Atif Azad
BDS Group
CSIS Department
University of Limerick, Ireland
atif.azad@ul.ie

ABSTRACT

The quality of candidate solutions in evolutionary computation (EC) depend on multiple independent runs and a large number of them fail to guarantee optimal result. These runs consume more or less equal or sometimes higher amount of computational resources on par the runs that produce desirable results.

This research work addresses these two issues (run quality, execution time), Run Prediction Model (RPM), in which undesirable quality evolutionary runs are identified to discontinue from their execution. An Ant Colony Optimization (ACO) based classifier that learns to discover a prediction model from the early generations of an EC run.

We consider Grammatical Evolution (GE) as our EC technique to apply RPM that is evaluated on four symbolic regression problems. We establish that the RPM applied GE produces a significant improvement in the success rate while reducing the execution time.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search - Heuristic methods

Keywords

Grammatical Evolution, Ant Mining, Machine Learning, Symbolic Regression, Training Set.

1. INTRODUCTION

Achieving optimal solution quality is a serious concern for the stochastic evolutionary algorithms (EAs) that require multiple independent runs. For the computationally expensive problems, the resources are better utilized if the EA produces minimum number of poor solutions. For example [7, 8] are more concerned about producing a single useful solution than to make confirmations to mimic the same result. The aim of this research is to offer a new mechanism that helps GE to optimally utilize the available resources by predicting, there by culling the runs that tend to turnout as poor solutions. An ACO [3] induced prediction model is applied in devising the performance of a given GE run. Further more, ACO discovered models are scrutinized that improve the number of qualitative solutions while optimizing the resource utilization.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07
http://dx.doi.org/10.1145/2598394.2598471.

2. PRELIMINARIES

GE is an EA that evolves computer programs in an arbitrary language through Context Free Grammars (CFGs) that are specified in Backus Naur Form (BNF). In general, a number of solution quality improvements were proposed in Genetic Programming (GP), of which, [4] is one such important GP based improvement on symbolic regression that used interval arithmetic and linear scaling.

This study is more focused in employing an ACO based classifier named $cAntMiner_{PB}$ [6] for performance prediction of a GE run. It discovers a list of classification rules with the help of an *on-the-fly* discretization procedure that enhances the accuracy on numerical data. $cAntMiner_{PB}$ takes the training data as an input and iteratively discovers rules by building construction graphs that produce intermediate rules, prunes these rules and removes all the data points covered by these intermediate rules. This process is repeated until all the data points in the input data set are covered or a convergence criterion is met. A detailed description of these new classifiers can be found in the survey shown in [5].

3. THE RUN PREDICTION MODEL

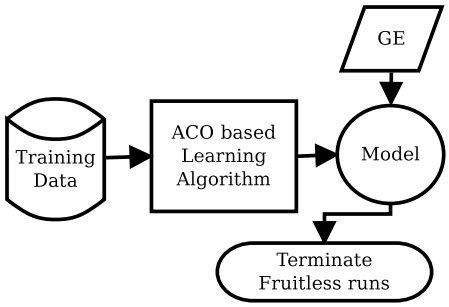

Figure 1: A block diagram of the run prediction model (RPM) applied GE.

The proposed run prediction model (RPM) follows the traditional design of GE except for a small tweak at the run

execution level. Figure 1 shows the block diagram of RPM applied GE. $cAntMiner_{PB}$ learns to discover a prediction model from the training data. The model accepts the GE evolutionary cycle as an input then, tries to identify the quality attainable at the end. Depending upon the model prediction, the runs that are predicted to be fruitless are terminated that results in continuation of the fruitful runs.

The training set required for the classifier holds the data that contains the respective changes in the parameters best fitness (BFC), average fitness (AFC), average actual length (AALC), average effective length (AELC) from generation 1 to 10.

4. EXPERIMENTS AND RESULTS

Our experiments are conducted in two sets on four symbolic regression problems. In the first set, the $cAntMiner_{PB}$ classifiers outperformed some state-of-the-art machine learning algorithms (C4.5, PART, JRip) in producing accurate prediction models on training data. The data collected from the experiments shown in [1] form the training data. The $cAntMiner_{PB}$ parameters are set to default as proposed in [6]. The classifier induces a list of classification rules that are of the form *IF BFC <= 0.0268 THEN no*. The discovered models are further analyzed to select the optimal combination of rules from the list of rules that are finally used to predict the solution quality of a given GE run.

In the second set of experiments, we compare the standard GE and, RPM applied GE on four symbolic regression problems in terms of the number of fruitful runs. The experimental parameters for GE: *# of runs* = 30, *population size* = 200, *# of generations* = 120, *crossover probability* = 0.9 (used effective crossover specific to GE), *mutation probability* = 0.01 (point mutation). randomly initialized the population with a *minimum genotype length* = 15, *maximum genotype length* = 25, *seed* is incremented in each iteration and, steady state replacement strategy were used. An *S-Lang* evaluator was used to evaluate the fitness of phenotype. We used normalized fitness as our fitness measure.

Table 1: Experimental results of standard GE vs RPM+GE over 30 runs.

Problem	GE	RPM+GE	
	fruitful runs	fruitful runs	culled runs
$f_1 = (1+x)^3$	18	15	9
$f_2 = x^4 - x^3 - y^2 - y$	7	3	23
$f_3 = x^3 - y^3 - y - x$	6	5	9
$f_4 = x^y$	518	12	13

Table 1 summarizes the standard GE results in comparison with the RPM applied GE across 30 independent runs. The results also contain the information regarding the number of discontinued runs as a result of applying the new technique. The proposed approach with Wilcoxon Signed Rank tests shows a significant improvement in the number of successful runs over standard GE at $\alpha = 0.05$. f_3 shows a significantly small improvement in its results whereas the remaining three problems report satisfactory improvements. Notice that f_2 shows an enormous increment in the number of successful runs that resulted from identifying many poor runs and culling them. There is a chance for the model to pluck the runs that actually turn out to yield qualitative solution but are predicted as poor runs. This can be expected as every predictor exerts certain amount of classification error. In fact, this is the prime reason for a small improvement in the case of f_3. Despite the inherent error rate of the new technique it shows significant improvements in its results.

We also focus on utilizing the available computational resources efficiently. The total execution time results of standard GE vs RPM+GE show a significantly high amount of reduction at $\alpha = 0.05$. At the end, it helped in optimal utilization of the GE computational resources.

5. CONCLUSION

This study has improved the solution quality of independent GE runs by applying a completely novel prediction approach that uses an accurate ACO classifier to devise a relatively simple prediction model. This also has significantly reduced the time spent on executing GE to attain the desirable quality of the solution. We focus on extending this in two steps: the first step will be to tackle the scaled up versions of the problems, in the next step we investigate in rapidly retraining the predictors as explained in [2, 8], that would result in highly accurate predictors.

6. REFERENCES

[1] G. Chennupati, C. Ryan, and R. M. A. Azad. An empirical analysis through the time complexity of GE problems. In R. Matousek, editor, *19th International Conference on Soft Computing, MENDEL'13*, pages 37–44, Brno, Czech Republic, jun, 26-28 2013.

[2] D. Costelloe and C. Ryan. On improving generalisation in genetic programming. In L. Vanneschi, S. Gustafson, A. Moraglio, I. Falco, and M. Ebner, editors, *Genetic Programming*, volume 5481 of *LNCS*, pages 61–72. Springer, Berlin, Heidelberg, 2009.

[3] M. Dorigo and T. Stützle. *Ant colony optimization*. MIT Press, 2004.

[4] M. Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, editors, *Genetic Programming*, volume 2610 of *LNCS*, pages 70–82. Springer, Berlin, Heidelberg, 2003.

[5] D. Martens, B. Baesens, and T. Fawcett. Editorial survey: swarm intelligence for data mining. *Machine Learning*, 82(1):1–42, 2011.

[6] M. Medland and F. E. B. Otero. A study of different quality evaluation functions in the cant-miner(pb) classification algorithm. In *GECCO*, pages 49–56, 2012.

[7] U.-M. O'Reilly, M. Wagy, and B. Hodjat. Ec-star: A massive-scale, hub and spoke, distributed genetic programming system. In R. Riolo, E. Vladislavleva, M. D. Ritchie, and J. H. Moore, editors, *Genetic Programming Theory and Practice X*, Genetic and Evolutionary Computation, pages 73–85. Springer New York, 2013.

[8] C. Ryan, M. Keijzer, and M. Cattolico. Favourable biasing of function sets using run transferable libraries. In U.-M. O'Reilly, T. Yu, R. Riolo, and B. Worzel, editors, *Genetic Programming Theory and Practice II*, volume 8 of *Genetic Programming*, pages 103–120. Springer US, 2005.