Multi-Sample Evolution of Robust Black-Box Search Algorithms

Matthew A. Martin
Natural Computation Laboratory
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri, U.S.A.
mam446@mst.edu

Daniel R. Tauritz
Natural Computation Laboratory
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri, U.S.A.
dtauritz@acm.org

ABSTRACT

Black-Box Search Algorithms (BBSAs) tailored to a specific problem class may be expected to significantly outperform more general purpose problem solvers, including canonical evolutionary algorithms. Recent work has introduced a novel approach to evolving tailored BBSAs through a genetic programming hyper-heuristic. However, that first generation of hyper-heuristics suffered from overspecialization. This poster paper presents a second generation hyper-heuristic employing a multi-sample training approach to alleviate the overspecialization problem. A variety of experiments demonstrated the significant increase in the robustness of the generated algorithms due to the multi-sample approach, clearly showing its ability to outperform established BBSAs. The trade-off between a priori computational time and the generated algorithm robustness is investigated, demonstrating the performance gain possible given additional run-time.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.2 [**Artificial Intelligence**]: Automatic Programming–*program synthesis*

General Terms

Algorithms, Design

Keywords

Black-Box Search Algorithms, Evolutionary Algorithms, Genetic Programming, Hyper-Heuristics

1. INTRODUCTION

Practitioners tend to be interested in solving a particular problem class which may fall anywhere on the continuum from a single instance problem to an arbitrarily large problem class. However, progress in the field of meta-heuristics has typically been aimed at solving increasingly varied problem classes. There is a clear need for meta-heuristics tunable to the needs of practioners in terms of the scope of the problem classes of interest, whether that be solving solely instances of MAXSAT with a fixed clause length and set number of variables, or arbitrary MAXSAT instances.

This paper introduces an improved version of the hyper-heuristic employing Genetic Programming (GP) introduced in [3] which drastically decreases the probability of evolving BBSAs that suffer from overspecialization. This improvement trains the evolved BBSAs employing a multi-sample approach. We present an investigation of the trade-off between the extra a priori computational time due to increasing sampling size and the increased robustness of the generated BBSAs in terms of lower variation in performance when varying the problem configuration. This is of critical importance to practitioners who need to be able to rely on the consistency of the generated BBSAs on all instances of their problem class of interest.

The goal of the research reported in this paper is to show that increasing the multi-sampling level increases the robustness of the generated BBSAs. Two primary measurements are employed to determine robustness [2]. The first is fallibility; if this value is large, it means that the BBSA can have a large difference in performance depending on problem configuration. The second measure is applicability: the size of the problem configuration space in which the BBSA performs better than a threshold value. For a BBSA to be highly robust, it should have a small fallibility and a large applicability.

This poster paper summarizes the experiments detailed in [4] where the related work and a detailed description of the methodology and experiments can be found.

2. METHODOLOGY

The specific focus of the research reported in this paper is to evolve robust BBSAs for a specific problem class using a multi-sample evaluation which can consistantly outperform more general purpose BBSAs. Koza-style tree-based GP was employed to evolve the algorithms where fitness was based on the performance averaged over a set of training problem configurations. The non-terminal nodes that compose these trees are operations extracted from preexisting algorithms and the terminal nodes are sets of solutions.

GP is employed to meta-evolve the BBSAs. Each individual in the GP population encodes a BBSA in a parse-tree. The fitness of a BBSA is estimated by computing the fitness

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2598448.

function that it employs on the solutions it evolves averaged over multiple runs. Each run of the BBSA begins with population initialization, followed by the parse-tree being repeatedly evaluated until one of the termination criteria is met.

The two primary variation operators employed are the standard sub-tree crossover and mutation altered to make the maximum number of nodes being added a user defined value. To ensure that the genetic program produces good BBSAs, the ones which do not evaluate any solutions are discarded upon generation.

A major issue identified in [3] is the problem of overspecialization when training on a single problem configuration of a given problem class. Following the approach suggested in [3], the BBSAs are executed on multiple problem configurations of the problem class of interest. On each problem configuration, the BBSAs run a user-specified number of times. This addition allows the user to control the robustness of the generated BBSA. If the user requires a BBSA that performs very consistently, then running the algorithm with more problem configurations is beneficial.

3. EXPERIMENTS

To demonstrate that the addition of multi-sampling evaluation of the BBSA reduces the probability of overspecialization, the algorithm was run on a series of multi-sampling levels, where a level is defined by the number of training problem configurations it samples. Once the BBSA has been evolved with a given multi-sampling level, it is tested on a superset of problem configurations to estimate the robustness of the BBSA and to demonstrate that it can outperform a standard EA.

The classic Deceptive Trap benchmark problem [1] was chosen to compare the results in this paper with those in [3], where BBSAs were evolved and suffered from overspecialization. The main purpose of the experiment was to study the effect of multi-sampling on the performance landscape across a wide set of problem configurations. The areas of interest in this experiment are the problem configurations that are very far away from the trained problem configurations. The BBSA with the largest fallibility, where fallibility indicates the difference between best and worst performance on the test problem configurations, were selected from each multi-sampling level to demonstrate a worst case scenario. These BBSAs, along with an EA, were run on all problem configurations with k from 4 to 20 inclusive and bit-lengths from roughly 70 to 500. The algorithms were run five times on each problem configuration.

4. DISCUSSION & CONCLUSION

The results presented show that the robustness improves as the multi-sampling level is increased. The least robust BBSA found using multi-sampling level one, performs well in the immediate area around the problem configuration that it was trained on. Unsurprisingly, as the problem configuration gets farther away from the trained problem configuration, the fitness decreases. This algorithm performs similarly to other algorithms that are tuned to specific problem configurations. When compared to how the EA performs on the same problem configuration space, the BBSA outperforms the EA in problem configurations near the trained problem configuration, but performs at near the same level as the distance increases. The variance in performance of the algorithm decreases as the multi-sampling level increases.

Training a BBSA on a larger number of training problem configurations improves the performance of the BBSA. In most cases, the improved performance is restricted to problem configurations that are relatively close to the trained problem configurations. However, when multi-sampling is built into the generation of the algorithm rather than solely the parameter tuning, the increased performance of the algorithm can be generalized to larger portions of the problem configuration space. The fallibility decreases as the multi-sampling level increase. Note that the training sets that these algorithms were generated on were restricted to a k from 5 to 7 and a $bit - length$ from 100 to 300 and the problem configuration space includes a k from 4 to 20 and a $bit - length$ from approximately 75 to 500. This method is shown to not only generate BBSAs that generalize to the problem configuration space close to the trained problem configurations, but to create BBSAs that have generalized to a much wider area of the problem configuration landscape. Though it is possible to evolve robust algorithms without using the multi-sampling method, it is shown that with a higher multi-sampling level, the general robustness of the evolved BBSA is increased along with the certainty that the evolved BBSA will indeed be robust.

One drawback of this method is the increased computational time that it requires. One cause of this increase are the additional runs that are necessary during the evaluation of a given BBSA. This extra computational time increases linearly with the multi-sampling level. It was noticed during testing and in the final results that the experiments run at a higher multi-sampling level can have a lower average fitness. Due to this result, a trend in the applicability is not statistically discernable. The BBSAs evolved at multi-sampling level five had the lowest trained fitness. This is believed to be caused by the increased difficulty of finding an algorithm that performs well on all of the training problem configurations. These two aspects cause the computational time increase of $\Omega(L)$, with L being the multi-sampling level.

5. REFERENCES

[1] K. Deb and D. Goldberg. Analyzing Deception in Trap Functions. In *Proceedings of FOGA II: the Second Workshop on Foundations of Genetic Algorithms*, pages 93–108, 1992.

[2] A. Eiben and S. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19 – 31, 2011.

[3] M. A. Martin and D. R. Tauritz. Evolving Black-box Search Algorithms Employing Genetic Programming. In *Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion*, GECCO '13 Companion, pages 1497–1504, New York, NY, USA, 2013. ACM.

[4] M. A. Martin and D. R. Tauritz. A Problem Configuration Study of the Robustness of a Black-Box Search Algorithm Hyper-Heuristic. In *Proceeding of the Sixteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion*, GECCO '14 Companion, New York, NY, USA, 2014. ACM.