Introducing Particle Swarm Optimization into a Genetic Algorithm to Evolve Robot Controllers

Malte Langosz
DFKI - Robotics Innovation Center Bremen
Robert-Hooke-Straße 1
28359 Bremen, Germany
malte.langosz@dfki.de

Kai A. von Szadkowski
University of Bremen
Robert-Hooke-Straße 1
28359 Bremen, Germany
kai.von-szadkowski@uni-bremen.de

Frank Kirchner
DFKI - Robotics Innovation Center / University of Bremen
Robert-Hooke-Straße 1
28359 Bremen, Germany
frank.kirchner@dfki.de

ABSTRACT

This paper presents Swarm-Assisted Behavior Graph Evolution (SABRE), a genetic algorithm which combines elements from genetic programming and neuroevolution to develop Behavior Graphs (BGs). SABRE evolves graph structure and parameters in parallel using particle swarm optimization (PSO) for the latter. The algorithm's performance was evaluated on a set of black-box function approximation problems, one of which represents part of a robot controller. We found that SABRE performed significantly better in approximating the mathematically complex test functions than the reference algorithms genetic programming (GP) and NEAT.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Learning*

Keywords

Genetic Algorithms, Genetic Programming, Neuroevolution, Patricle Swarm Optimization, Robotics

1. INTRODUCTION

In this work we present a new genetic algorithm - Swarm-Assisted Behavior Graph Evolution (SABRE) - and vary its components to explore two ways of improving current evolutionary methods: (I) changing the properties of the structural elements and (II) applying a specialized algorithm for tuning the structures' parameters. Behavior Graphs (BGs) were recently presented as an approach combining the function representation of artificial neural networks (ANNs) and genetic programming (GP) to allow the construction of simple network structures capable of complex calculations [**?**]. As for point (II), the potential of local parameter search on evolving structures has previously been shown [**?**]. Particle Swarm Optimization (PSO) is a widely known search algorithm which performs comparably well or better than other methods especially on complex or changing fitness landscapes.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2598474.

2. GENETIC BEHAVIOR GRAPHS

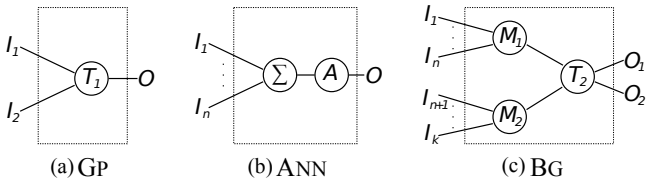
SABRE uses a genetic representation to evolve BG structures and their parameters in parallel. The former are developed using a genetic algorithm implementation, while the latter are subjected to PSO.

A behavior node combines the node representation used in GP with the representation used in ANNs (Fig. **??**). Possible merge functions are weighted sum (`SUM`), product (`PRODUCT`), the signal with the greatest weight (`WTA`) as well as `MAX` and `MIN`. The mapping between the inputs and outputs is done by a *transfer function.* Possible functions include `PIPE` which simply maps the input to the output, the mathematical functions `DIV`, `SIN`, `COS`, `TAN`, `ACOS`, `ATAN2`, `POWER` and `SIGMOID`, as well as `PHI` which facilitates conditional branching. Most of these functions take a single input and provide a single output, except `ATAN2` and `POWER` which take two inputs and `PHI` which takes three.

In SABRE, BGs are represented by gene strings containing genes encoding nodes and connections. Newly developed genes are attached to the end of the gene string, thereby sorting the genes by age. Two nested loops are implemented, the outer one dealing with structure development, running continuesly, and the inner optimizing the parameters of the gene strings, iterating n times per outer loop iteration. To realize this hierarchical subdivision, the population is split into subpopulations P_1 through P_κ, whereby each subpopulation P_i represents one structure. What varies within subpopulations (consisting of λ individuals), are the parameters defined in the genes. Two possibilities for varying gene parameters were used: PSO, the standard for SABRE, and a $1 + \lambda$ evolutionary strategy (ES). For the experiments described here, n, κ and λ were set to 10.
Whenever a new gene is created, it is assigned a unique identifier – similar to the historical markers used in NEAT [**?**]. In the PSO implementation, every particle of the swarm is given one segment per unique gene it maps to (the segment encoding as many parameters as required by the particular gene). The unique identifiers of the genes ensure correct mapping even when gene strings are cloned or of varying sizes.

3. EXPERIMENTS

Black-box function approximations were used to evaluate the performance of SABRE. In total, four variants of



(a) GP (b) ANN (c) BG

Figure 1: Different representations of nodes used in a GP tree (a), ANN such as used in NEAT (b), and BGs (c). A GP node defines a fix number of inputs and one output to be used in the tree structure of the GP phenotype (e.g. node T_1 gets two inputs). The ANN node uses a transfer function $\sum$ to merge a variable number of inputs to one value that is used by an activation function A. The activation function calculates the output of the node. The BG node mixes both concepts by defining a transfer function T_2 with a fixed number of inputs. The transfer function can also have more than one output. For each input a merge function M_1 and M_2 is defined.

SABRE were tested, using a full set of transfer functions (SABRE_{full}) and only the sigmoid function (SABRE_{σ}) both with and without PSO: $\text{SABRE}_{full/PSO}$, $\text{SABRE}_{full/ES}$, $\text{SABRE}_{\sigma/PSO}$, $\text{SABRE}_{\sigma/ES}$. A standard GP algorithm as well as the neuroevolutionary algorithm NEAT were used for performance comparison. Similar to SABRE, GP was used in two configurations: with a full and a reduced (only `SIG` and `ADD`) set of transfer functions. Each of the combinations of algorithm and benchmark function were replicated 100 times on a training set of inputs drawn randomly from a uniform distribution $\mathcal{U}[-1,1]$. In every such setting, the chosen algorithm was allowed to compute 500,000 fitness evaluations. Fitness was computed as root-mean-square error (thus smaller values meaning higher fitness). It was evaluated on the training set, however, fitness values reached on an independently created equally-sized test set were used to track fitness development over time and for comparison of fitness distributions between the algorithms. Two benchmark functions were used: f_1 and f_2. Function f_1 is a combination of nested sigmoid functions and sums and can be represented by ANNs with three input nodes, two hidden nodes and two output nodes. The function is defined as:

$$f_{1_{out1}}(x_1,x_2,x_3) = \sigma(0.1\sigma(0.25x_1 + 0.4x_2 - 1.2) + 0.3\sigma(x_2 + x_3 - 0.215) + 0.1123)$$
$$f_{1_{out2}}(x_1,x_2,x_3) = \sigma(x_1 + 0.0314)$$

Function f_2 represents the inverse kinematics calculation of a leg of the SpaceClimber robot [**?**] and possesses four inputs and four outputs.

4. RESULTS & DISCUSSION

For both benchmark functions, achieved fitness on the test set was significantly different between the examined algorithms (Mann–Whitney U-test, $p < 5\%$), with $\text{SABRE}_{\sigma/PSO}$ performing best, followed by $\text{SABRE}_{full/PSO}$ and NEAT (Fig. **??**). $\text{SABRE}_{\sigma/ES}$ and $\text{SABRE}_{full/ES}$ show intermediate performance in both functions. With both ES and Pso, SABRE performs better with a reduced function set (only `SIGMOID`). GP_{sigma} performed worst in both tests, while GP_{full} was almost as good as NEAT and $\text{SABRE}_{full/PSO}$ in f_1.

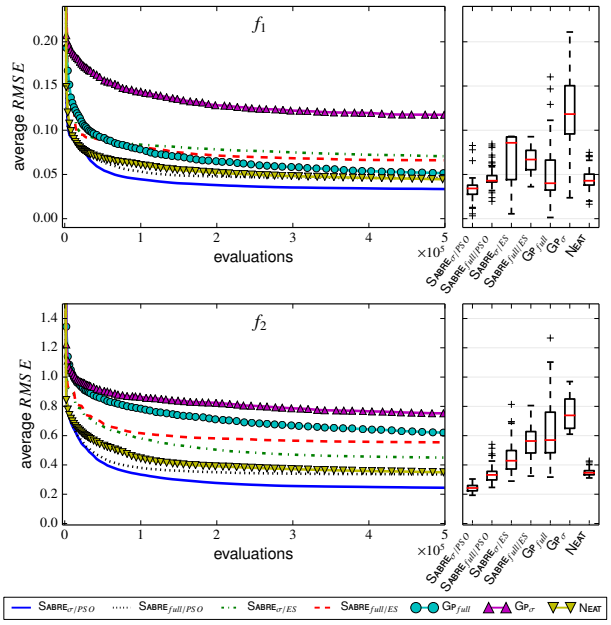


Figure 2: Development of average $RMSE$ over runtime; corresponding final $RMSE$ distribution is displayed as boxplots.

The results indicate that SABRE in its standard form using PSO outperforms the variants using ES. Thus using PSO to optimize parameters BGs seems a very promising approach. It is noteworthy that SABRE performed better with a reduced function set, which may be explained by the comparably small incremental steps and thus the evolutionary stability resulting from the use of a sigmoid transfer function only.

As for the overall performance of SABRE, we can conclude from our data that for certain types of complex problems such as inverse kinematics of real robots, SABRE is able to produce superior results compared to standard GP and NEAT. It has to be noted that it may be possible to find configuration parameters for the two comparison algorithms changing these results, however we were unable to obtain such parameter sets despite various tests.

5. REFERENCES

[1] S. Bartsch, T. Birnschein, M. Römmermann, J. Hilljegerdes, D. Kühn, and F. Kirchner. Development of the six-legged walking and climbing robot spaceclimber. *Journal of Field Robotics*, pages 506–532, 2012.

[2] Y. Kassahun and G. Sommer. Efficient reinforcement learning through evolutionary acquisition of neural topologies. In *13th European Symposium on Artificial Neural Networks (ESANN)*, 2005.

[3] M. Langosz, L. Quack, A. Dettmann, S. Bartsch, and F. Kirchner. A behavior-based library for locomotion control of kinematically complex robots. *Proceedings of the 16th International Conference on Climbing and Walking Robots, (CLAWAR)*, 2013.

[4] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.