

Instance-Based Parameter Tuning for Evolutionary AI Planning

Mátyás Brendel
Projet TAO, INRIA Saclay & LRI
Université Paris Sud
Orsay, France
matthias.brendel@lri.fr

Marc Schoenauer
Projet TAO, INRIA Saclay & LRI
Université Paris Sud
Orsay, France
marc.schoenauer@inria.fr

ABSTRACT

Learn-and-Optimize (LaO) is a generic surrogate based method for parameter tuning combining learning and optimization. In this paper LaO is used to tune Divide-and-Evolve (DaE), an Evolutionary Algorithm for AI Planning. The LaO framework makes it possible to learn the relation between some features describing a given instance and the optimal parameters for this instance, thus it enables to extrapolate to unknown instances in the same domain. Moreover, the learned model is used as a surrogate-model to accelerate the search for the optimal parameters. The proposed implementation of LaO uses an Artificial Neural Network for learning the mapping between features and optimal parameters, and the Covariance Matrix Adaptation Evolution Strategy for optimization. Results demonstrate that LaO is capable of improving the quality of the DaE results even with only a few iterations. The main limitation of the DaE case-study is the limited learning time and amount of meaningful features that are available. However, it is demonstrated that the learned model is capable of generalization in the domain to unknown instances.

Categories and Subject Descriptors

I.2.6 [Computing Methodologies]: Artificial Intelligence Learning Parameter learning

General Terms

Theory

Keywords

parameter tuning, AI Planning, evolutionary algorithms

1. INTRODUCTION

All state of the art parameter tuning techniques, like for example F-Race, REVAC and ParamILS [5] face the same crucial generalization issue: the generalization of a parameter-set that has been optimized for a given problem to another one. The answer of course depends on the similarity of the problems. However, in AI Planning, sufficiently accurate features have not been specified that would allow to describe the problem, no design of a general learning framework has

been proposed, and no general experiments have been carried out. This paper makes a step toward a framework for parameter tuning applied generally for AI Planning and proposes a preliminary set of features. The Learn-and-Optimize (LaO) framework consists of the combination of optimizing and learning, i.e., finding the mapping between features and best parameters.

In this paper, the target optimization technique is Evolutionary Algorithms (EA), more precisely the evolutionary AI planner called Divide-and-Evolve (DaE). However, DaE will be here considered as a black-box algorithm as described in [4].

2. LEARN-AND-OPTIMIZE

As [1] demonstrates parameters tuned for one instance may not be optimal for other instances. One workaround is to aim at learning a complex relation between instances and optimal parameters.

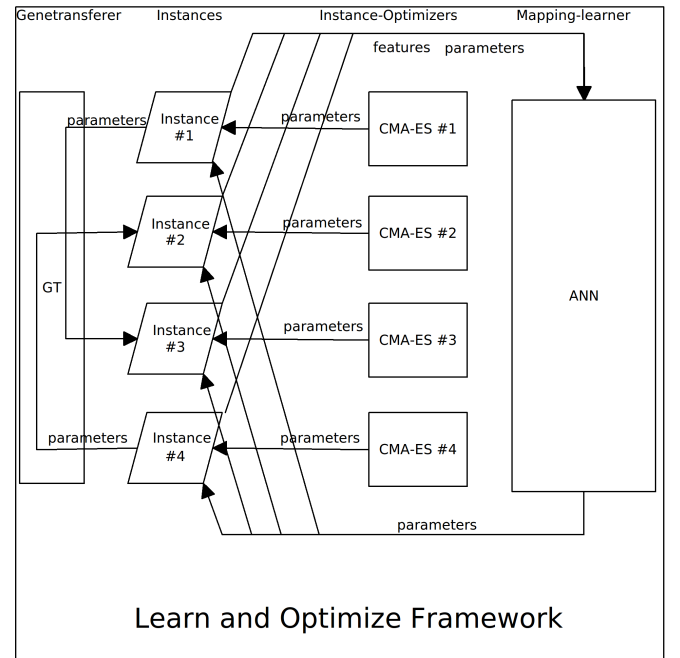


Figure 1: Flowchart of the LaO framework, displaying only 4 instances.

Domain Name	# of iterations	# training instances	# test instances	ANN error	quality-ratio in LaO	quality-ratio ANN on train	quality-ratio ANN on test
Freecell	16	108	230	0.1	1.09	1.05	1.04
Grid	10	55	124	0.09	1.09	1.05	1.03
Mprime	8	64	152	0.08	1.11	1.05	1.04

Table 1: Results by domains. ANN-error is given as MSE, as returned by FANN. The quality-improvement ratio "in LaO" is that of the best parameter-set found by LaO.

Suppose that we have n features and m parameters (considered to be continuous), and we are doing per-instance parameter tuning on instance \mathcal{I} by optimizing the fitness function $f_{\mathcal{I}} : \mathbf{R}^m \rightarrow \mathbf{R}$, the expected value of DaE executed with parameters $p \in \mathbf{R}^m$. The optimal parameter set is defined by $p_{opt} = \operatorname{argmin}_p \{f_{\mathcal{I}}(p)\}$. For each instance \mathcal{I} , consider the set of features: $F(\mathcal{I})$. We try to learn the mapping from the feature space to the optimal parameter space, which might not be totally unambiguous.

$$p : \mathbf{R}^n \rightarrow \mathbf{R}^m, p(F) = p_{opt} \quad (1)$$

A multilayer Feed-Forward Artificial Neural Network (ANN) was chosen for the learning of the features-to-parameters mapping. The intermediate models were already used in optimization as in standard surrogate-model based techniques. As an optimizer for each instance a simple yet robust (1+1)-Covariance Matrix Adaptation Evolution Strategy [2] was chosen. In addition random gene-transfer between instances was developed, transferring to all of the instances with uniform random distribution the so-far best parameter of a different instance. Figure 1 shows the LAO framework with the described implementations. The implementation of LaO uses the Shark library [3] for CMA-ES and the FANN library for ANN [6]. Since DaE is not deterministic, the median of 11 independent runs of DaE with a given parameter-set was computed.

Name	Min	Max	Default
Probability of crossover	0.0	1	0.8
Probability of mutation	0.0	1	0.2
Rate of mutation add station	0	10	1
Rate of mutation delete station	0	10	3
Rate of mutation add atom	0	10	1
Rate of mutation delete atom	0	10	1
Mean average for mutations	0.0	1	0.8
Time interval radius	0	10	2
Maximum number of stations	5	50	20
Maximum number of nodes	100	100 000	10 000
Population size	10	300	100
Number of offspring	100	2 000	700

Table 2: DaE parameters that are controlled by LaO

One iteration of LaO amounts to 5 iterations of CMA-ES, followed by one ANN training and one Genetransferer. The ANN had 3 fully connected layers, the input and hidden layer had both 12 neurons corresponding the number of features. In one iteration of LaO, the ANN was only trained for 50 epochs. After stopping LaO, retraining was made with 300 ANN epochs with the best data, to avoid under-training. The controlled parameters of DaE are described in table 2. The feature-set consists of 12 features. The first 6 features are computed from the domain and instance files:

number of fluents, goals, predicates, objects and types. One further feature is called mutex-density, which is the number of so-called "mutexes" divided by the number of all fluent-pairs. We also kept 6 less important features: number of lines, words and byte-count of the instance and the domain file.

For evaluation the quality-improvement the quality-ratio metric defined in IPC competitions was used. The baseline is the default parameter-setting. The ratio of the fitness value for the default parameter and the tuned parameter was computed and average was taken over the instances in the train or test-set.

$$Q = \frac{Fitness_{baseline}}{Fitness_{tuned}} \quad (2)$$

3 sample domains were selected from the Planning and Learning Part of IPC2011: Freecell, Grid, and Mprime. As it can be seen in Table 1, some quality-gain in training was consistently achieved, but the transfer of this improvement to the ANN-model was only partial, probably because the feature-set is not strong enough. On the other hand, the ANN model generalizes excellently to the the independent test-set.

3. REFERENCES

- [1] J. Bibai, P. Savéant, M. Schoenauer, and V. Vidal. On the generality of parameter tuning in evolutionary planning. In J. B. et al., editor, *Genetic and Evolutionary Computation Conference (GECCO)*, pages 241–248. ACM Press, July 2010.
- [2] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
- [3] C. Igel, T. Glasmachers, and V. Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.
- [4] Jacques Bibai, Pierre Savéant, Marc Schoenauer, and Vincent Vidal. On the benefit of sub-optimality within the divide-and-evolve scheme. In P. Cowling and P. Merz, editors, *EvoCOP 2010*, number 6022 in Lecture Notes in Computer Science, pages 23–34. Springer-Verlag, 2010.
- [5] E. Montero, M.-C. Riff, and B. Neveu. An evaluation of off-line calibration techniques for evolutionary algorithms. In *Proc. ACM-GECCO*, pages 299–300. ACM, 2010.
- [6] N. Nissen. Implementation of a Fast Artificial Neural Network Library (FANN). Technical report, Department of Computer Science University of Copenhagen (DIKU), 2003.