Levels of Abstraction in Modeling and Sampling: The Feature-Based Bayesian Optimization Algorithm

Moshe Looks
Washington University in St. Louis
Dept. of Computer Science and Engineering
One Brookings Dr., St. Louis, MO 63130
mlooks@cs.wustl.edu

ABSTRACT

I introduce a generalization of probabilistic modeling and sampling for estimation of distribution algorithms (EDAs), that allows models to contain *features*, additional level(s) of abstraction defined in terms of the problem's base-level variables. I demonstrate how a simple feature class, variable-position motifs within fixed-length strings, may be exploited by a powerful EDA, the Bayesian optimization algorithm (BOA). Experimental results are presented where motifs are learned autonomously via a simple heuristic. The effectiveness of this feature-based BOA is demonstrated across a range of problems where such motifs are relevant.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation

Keywords

Empirical Study, Heuristics, Optimization, Representations

1. MOTIVATION AND METHOD

An Estimation of Distribution Algorithm (EDA) builds a probabilistic model over the space of solutions to an optimization problem, and generates new solutions by sampling from the probability distribution encoded by its model - that is, generating assignments for a set of random variables. But where do these random variables come from? Typically, they correspond to a prespecified encoding (e.g., the characters in a fixed-length string). In this paper, I describe how the modeling and sampling stages for EDAs may be generalized to contain an added level of abstraction, *features*.

Linkage-learning EDAs mine population-level statistics to uncover interactions between variables - a feature-based EDA may also uncover interactions between features. To illustrate, I show how a simple feature class, variable-position motifs within fixed-length strings, may be exploited by the Bayesian optimization algorithm (BOA) [2], creating the feature-based BOA (fbBOA).

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

Assume learning over the bits in a fixed-length binary string, $(X_1, X_2, ..., X_n)$ - I will refer to these henceforth as "base-level" variables.[1] The BOA models a population as a Bayesian network among these variables. To sample new solutions, nodes of the network are ordered topologically, and values are sequentially assigned to variables according to the conditional probabilities encoded by their parents'.

Henceforth let a feature be some well-defined Boolean predicate over the space of solutions - for modeling purposes, such features may simply be added as random variables to the Bayesian network (since base-level variables are a particular kind of feature, this is a generalization of standard modeling). However, sampling from a distribution subject to an arbitrary predicate is a hard problem in general. The main contribution of this paper is a heuristic to use for this kind of sampling in a population-based learning context.

A feature's *grounding* may be defined with respect to (wrt) a particular solution x - some set of variables S will be said *fix a feature on* x if the feature is present in every solution with the same assignments as x for all variables in S. Such a set is *minimal* if none of its subsets fix the feature on x. A feature's grounding wrt a solution x may now be defined as the union of all minimal sets of variables which fix it on x. By this definition, the grounding of a feature wrt a solution where the feature is absent is the empty set.

To illustrate, consider the feature "contains the substring 11", defined over length-8 binary strings $(X_1, X_2, ..., X_8)$. The grounding wrt 10110111 is $\{X_3, X_4, X_6, X_7, X_8\}$, since the minimal fixing sets for this string are $\{X_3, X_4\}$, $\{X_6, X_7\}$, and $\{X_7, X_8\}$. The definition of a grounding allows us, given a solution x where a feature is present, to generate new solution where the feature is present *in the same way that it is present in* x, which is important given that there may be many ways for a feature to be present, in contrast to assignment of a variable, which may only be carried out in a single way. Setting a feature to true thus corresponds to setting all variables in the grounding of some solution - if this is not possible another solution is drawn, without replacement (note that this process may fail). To set a feature to false, the distribution is transformed by eliminating the groundings of *all* solutions with the feature from consideration (this fails if doing so would reduce the set of possibilities to empty). If assigning a feature fails, it is simply skipped.

To demonstrate feature-based modeling and instance gen-

[1]Note that generalizations to other structures, including variable-sized and shaped program trees [1], are possible, and are not incompatible with the methods presented here.

eration, it is necessary to define some features. I will illustrate with the exploitation of useful motifs (repeated substrings), which are identified using a simple randomized search combined with a heuristic scoring function. These are learned dynamically - a number of random substrings (corresponding to the size of the population) are uniformly sampled and admitted if they score above a threshold (0.4 in all experiments). The scoring metric is the product of two factors, information gain (based on the frequency of a motif in the population) and spread (based on the frequency of a motif *across* different positions where it might occur). Information gain is conditional on the features already created, which greatly reduces the number of features created at a time (typically to five or less in the experiments).

2. EXPERIMENTS AND CONCLUSIONS

The simplest of the test problems is OneMax, where fitness is given by Hamming distance to the string of all ones. A more difficult problem, 3-deceptive, may be obtained by considering groups of three successive variables to form a "trap", leading away from the global optima (details in [2]).

Other problems considered are the TwoMax, or Twin-Peaks problem, which has two global optima (all ones and all zeros), and a global minimum when the string is half ones and half zeros. A more challenging synchronization problem is the one-dimensional Ising model, where the string is considered to be a ring, and fitness is subtracted for each transition between a one and a zero in successive elements (TwoMax and the Ising model are described in detail in [3]).

A third set of problems which incorporate hierarchical difficulty are are hierarchical if-and-only-if (H-IFF), and hierarchical exclusive-or (H-XOR), described in [4]. It is important to note that H-XOR represents a problem where repetition is not strongly present in the optima.

Results are shown in the figures. In these experiments, model-building used Bayesian networks with local structure (decision trees), learned via the Bayesian-Dirichlet metric - see [2] for details. Following the methodology used in [2], the population size is set empirically for each algorithm, problem, and size, to be the minimum necessary in order to achieve convergence to the optimum in all 30 independent runs. All other BOA-related parameters are fixed (for the BOA and the fbBOA) as in [2]. When running the fbBOA, substring features were learned after each generation.

In conclusion, I have described how the modeling and sampling which takes place in an EDA may be generalized to encompass abstract *features*, and presented experimental results to demonstrate the method's effectiveness. I am currently using this work as a springboard to develop EDAs for learning computer programs that exploit domain knowledge and operator semantics to reduce the search space, in combination with techniques developed in [1].

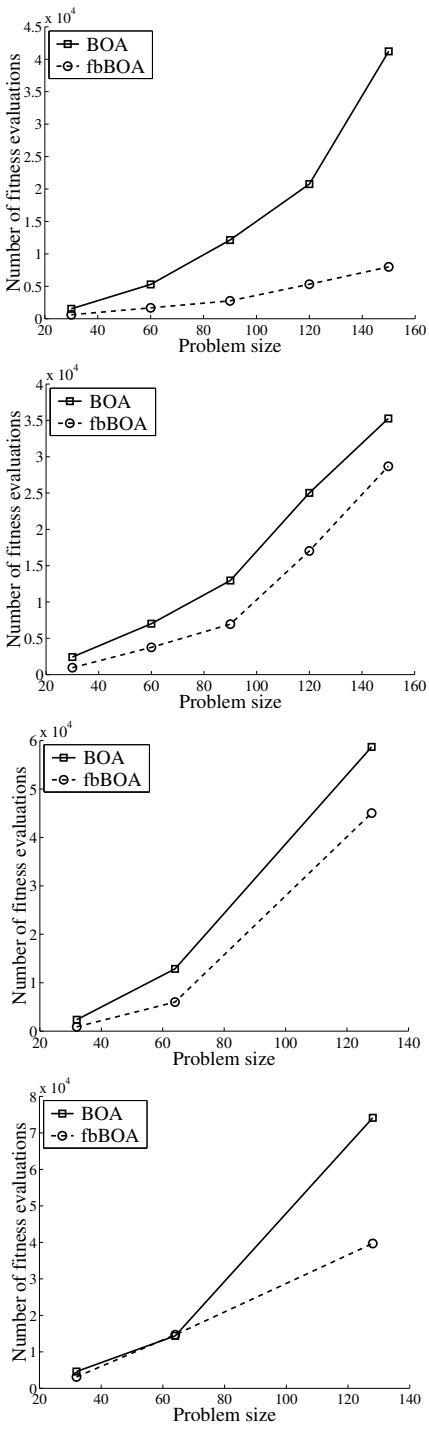

Figure 1: Performance on TwoMax, Ising, H-IFF, and H-XOR. Results for OneMax and 3-deceptive were qualitatively similar to TwoMax.

3. REFERENCES

[1] M. Looks, B. Goertzel, and C. Pennachin. Learning computer programs with the Bayesian optimization algorithm. In *GECCO*, 2005.

[2] M. Pelikan. *Bayesian Optimization Algorithm: From Single Level to Hierarchy.* PhD thesis, University of Illinois at Urbana-Champaign, 2002.

[3] C. Van Hoyweghen, D. E. Goldberg, and B. Naudts. From TwoMax to the Ising model: Easy and hard symmetrical problems. In *GECCO*, 2002.

[4] R. A. Watson, G. S. Hornby, and J. B. Pollack. Modeling building-block interdependency. In *Parallel Problem Solving from Nature (PPSN-V)*, 1998.