Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms

Frank Hutter[1], Youssef Hamadi[2], Holger H. Hoos[1], and Kevin Leyton-Brown[1]

[1] University of British Columbia, 2366 Main Mall, Vancouver BC, V6T1Z4, Canada
{hutter, kevinlb, hoos}@cs.ubc.ca
[2] Microsoft Research, 7 JJ Thomson Ave, Cambridge, UK
youssefh@microsoft.com

Abstract. Machine learning can be used to build models that predict the run-time of search algorithms for hard combinatorial problems. Such *empirical hardness models* have previously been studied for complete, deterministic search algorithms. In this work, we demonstrate that such models can also make surprisingly accurate predictions of the run-time distributions of incomplete and randomized search methods, such as stochastic local search algorithms. We also show for the first time how information about an algorithm's parameter settings can be incorporated into a model, and how such models can be used to automatically adjust the algorithm's parameters on a per-instance basis in order to optimize its performance. Empirical results for Novelty$^+$ and SAPS on structured and unstructured SAT instances show very good predictive performance and significant speedups of our automatically determined parameter settings when compared to the default and best fixed distribution-specific parameter settings.

1 Introduction

The last decade has seen a dramatic rise in our ability to solve combinatorial optimization problems in many practical applications. High-performance heuristic algorithms increasingly exploit problem instance structure. Thus, knowledge about the relationship between this structure and algorithm behavior forms an important basis for the development and successful application of such algorithms. This has inspired a large amount of research on methods for extracting and acting upon such information. These range from search space analysis to automated algorithm selection and tuning methods.

An increasing number of studies explore the use of machine learning techniques in this context [15,18,6,8]. One recent approach uses linear basis function regression to obtain models of the time an algorithm will require to solve a given problem instance [19,21]. These so-called *empirical hardness models* can be used to obtain insights into the factors responsible for an algorithm's performance, or to induce distributions of problem instances that are challenging for a given algorithm. They can also be leveraged to select among several different algorithms for solving a given problem instance.

In this paper, we extend on this work in three significant ways. First, past work on empirical hardness models has focused exclusively on complete, deterministic algorithms [19,21]. Our first goal is to show that the same methods can be used to predict sufficient statistics of the run-time distributions (RTDs) of incomplete, randomized algorithms, and in particular of stochastic local search (SLS) algorithms for SAT. This is

F. Benhamou (Ed.): CP 2006, LNCS 4204, pp. 213–228, 2006.

important because SLS algorithms are among the best existing techniques for solving a wide range of hard combinatorial problems, including hard subclasses of SAT [14].

The behavior of many randomized heuristic algorithms is controlled by parameters with continuous or large discrete domains. This holds in particular for most state-of-the-art SLS algorithms. For example, the performance of WalkSAT algorithms such as Novelty [20] or Novelty$^+$ [12] depends critically on the setting of a noise parameter whose optimal value is known to depend on the given SAT instance [13]. Understanding the relationship between parameter settings and the run-time behavior of an algorithm is of substantial interest for both scientific and pragmatic reasons, as it can expose weaknesses of a given search algorithm and help to avoid the detrimental impact of poor parameter settings. Thus, our second goal is to extend empirical hardness models to include algorithm parameters in addition to features of the given problem instance.

Finally, hardness models could also be used to automatically determine good parameter settings. Thus, an algorithm's performance could be optimized for each problem instance without any human intervention or significant overhead. Our final goal is to explore the potential of such an approach for automatic per-instance parameter tuning.

In what follows, we show that we have achieved all three of our goals by reporting the results of experiments with SLS algorithms for SAT. (We note however, that our approach is by no means limited to SLS algorithms or SAT, though the features we use were created with some domain knowledge. In experimental work it is obviously necessary to choose *some* specific domain. We have chosen to study the SAT problem because it is the prototypical and best-studied $\mathcal{NP}$-complete problem and there exists a great variety of SAT benchmark instances and solvers.) Specifically, we considered two high-performance SLS algorithms for SAT, Novelty$^+$ [12] and SAPS [17], and several widely-studied structured and unstructured instance distributions. In Section 2, we show how to build models that predict the sufficient statistics of RTDs for randomized algorithms. Empirical results demonstrate that we can predict the median run-time for our test distributions with surprising accuracy (we achieve correlation coefficients between predicted and actual run-time of up to 0.995), and that based on this statistic we can also predict the complete exponential RTDs Novelty$^+$ and SAPS exhibit. Section 3 describes how empirical hardness models can be extended to incorporate algorithm parameters; empirical results still demonstrate good performance for this harder task (correlation coefficients reach up to 0.98). Section 4 shows that these models can be leveraged to perform automatic per-instance parameter tuning that results in significant reductions of the algorithm's run-time compared to using default settings (speedups of up to two orders of magnitude) or even the best fixed parameter values for the given instance distribution (speedups of up to an order of magnitude). Section 5 describes how Bayesian techniques can be leveraged when predicting run-time for test distributions that differ from the one used for training of the empirical hardness model. Finally, Section 6 concludes the paper and points out future work.

2 Run-Time Prediction: Randomized Algorithms

Previous work [19,21] has shown that it is possible to predict the run-time of deterministic tree-search algorithms for combinatorial problems using supervised machine learning techniques. In this section, we demonstrate that similar techniques are able