

# Harnessing Algorithm Bias in Classical Planning

Mark Roberts

mroberts@cs.colostate.edu  
http://www.cs.colostate.edu/~mroberts  
Computer Science Department, Colorado State University  
Fort Collins, CO, 80521

A planning system's performance is biased due to many factors related to its design. For example, the representation, decision points, search control, memory usage, heuristic guidance, and stopping criteria all can have implications for performance. Problem instance characteristics also impact system performance. The interaction of the design choices with the problem instance makes it difficult to select the most efficient system from the array of choices. It seems natural to apply learning to the *algorithm selection* problem of allocating computational resources among a *portfolio* of planners that may have complementing (or competing) search technologies. Such selection is called the *portfolio strategy*.

Our working thesis is that *we can study a portfolio of planning systems for clues about why one algorithm is favored over another*. A secondary thesis is that *we can uncover algorithmic and problem structure dependencies by examining algorithm performance on specific instances*. Our research analyzes performance of planning systems by:

1. Modeling planner performance on benchmark problems.
2. Constructing portfolio strategies using a principled methodology based on analysis of and learning from previous off-line performance.
3. Measuring portfolio strategies to test specific hypotheses about what constitutes effective selection, ranking, and allocation.
4. Examining planner performance, models of the performance, features used to build those models, domain information, and dependencies between these sets to develop and test specific hypotheses leading to stronger explanations of search performance in planning.
5. Constructing a single planner that demonstrates the validity of our findings.
6. Supplementing the current benchmarks with problems that exploit our modeling knowledge.

What follows are highlighted findings for items (1) and (2) from last year's publications (Roberts 2006) (Roberts & Howe 2006) as well as the current state of items (3) and (4); the author's dissertation research will provide detailed analyses for all of these items.

## Initial Models and Portfolio

We started with 21 classical planners. For each planner, we constructed two models: *success* and *time*. Success models output a binary decision (succeed or fail) and usually estimate the probability of finding a solution given a problem instance and a planner ( $P(\text{solution found}|\text{problem, planner})$ ). The time models predict computation time needed for a given planner to complete a given problem.

**Features** Each problem instance is defined by 38 features automatically extracted from problem and domain definitions; we included features from (Howe *et al.* 1999) plus many others. We divided the features into three categories of increasing knowledge and computational cost: domain specific, instance specific, action interaction. We also examined 20 features from Hoffmann's (2004) state space topology analysis, but these features are intractable for realistic problem sizes<sup>1</sup>. Based on the amount of computation time to compute features, we designated the domain and instance-specific features as 'fast' and the action interaction and topological features as 'expensive' features.

**Problems** We started by running all planners on 3959 STRIPS PDDL problems from 77 domains. The problems are taken from Hoffmann's dataset (2004), the UCPOP Strict benchmark, IPC sets (IPC1, IPC2, IPC3 Easy Typed IPC4 Strict Typed) and 37 other problems from two domains (Sodor and Stek) that have been made publicly available. Each planner is allowed 30 minutes and 768 MB memory. We used 30 identically configured Pentium 4 3.4Ghz computers running Fedora Core 6.

**Models** We built models with the WEKA data mining package (Witten & Frank 2005); to start, we used two models that worked well that we could explain: OneR and J48. OneR selects the single feature that yields the highest prediction value on the training set, while J48 is a decision tree method based on Quinlan's C4.5. By default, we used 10-fold cross validation. The models are distinguished based on the data we use to build them: *all* data and *old* data (all but IPC4). We found that:

- When predicting success, J48 achieved 96.7% average accuracy (sd of 3.2) for *old* and 96.8% average accuracy (sd of 2.12) for *all*.

<sup>1</sup>We thank Jörg Hoffmann for supplying the code.

- When predicting time, the average accuracy using J48 on *log binned data* was 95.0% (sd of 2.49). Over all planners, 84.2% of the runs finish in less than one second and 5% finished in greater than 1000 seconds.
- Expensive features improved accuracy but not enough to justify their cost.
- Models trained on *old* did not generalize well to IPC4.
- Using OneR models to model success revealed that the average number of negations in effects was the best predictor for nine of the planners; the predicate arity was best for another four. The first feature may indicate where the often used  $h^+$  heuristic may have trouble; the second roughly influences branching in the search space.

**Initial Portfolio** Our portfolio accepts a new problem, *ranks* the available algorithms, then *allocates* computation time to them; it may also *prune* planners predicted to fail if given this information. We have compared this simple portfolio to the most successful and average planners and found:

- We can reduce the set of planners to those which are not dominated by another planner. Using a set-covering algorithm reduces the number of planners by about half.
- A simple allocation strategy that uses the run-time distributions can produce performance better than the average planner performance.

## Recent Extensions

To this initial examination we have added numerous extensions. We added 9 more planners (but two were not reliable) as well as 767 more challenging problems from the IPC5 Propositional and IPC4 hard-typed domains. We also added 30 more classifiers<sup>2</sup>. Finally, we modified the portfolio to use a variety of ranking and allocation strategies (including random baselines) and the ability to run algorithms either serially (in order by ranking) or in a round-robin fashion until success or time runs out. We found that:

- The expanded model set improves accuracy over the initial models; especially for runtime. The most common model was KStar – a variant of K-Nearest Neighbors.
- The new models generalize to newer problems when trained on older problems.
- The best ranking strategies employ the extended models.
- We need even more accurate time models; a random allocation strategy performed as well as any other strategy.
- Performance trends suggest that round-robin strategies are more successful than serial strategies.
- The best portfolio lags 60 seconds on average behind an ‘oracle’ selection strategy that uses perfect knowledge.

## Next Steps

There remains much work to *link* the search bias of various planners with their performance on specific problems. We present key points of continuing work we hope will identify dependencies between the domains, heuristics, algorithms, and runtime dynamics in classical planning.

<sup>2</sup>Another graduate student, Landon Flom, is applying the advanced Machine Learning techniques

We are currently examining performance interactions based on planner heuristic (Relaxed Graphplan (RGP) Heuristic vs. non-RGP heuristic). Further interactions related planner technologies (SAT-based, POCL-base, Graphplan, etc.) will be explored, which may also lead to limited parameter tuning of the planners.

The OneR and J48 models provide some evidence linking particular features to performance prediction. We expect to identify new features that help explain indirect action interactions as we perform richer domain analysis. Feature cost and feature selection are important to further enhancing the portfolio models. Finally, we will mine relationships between the features, the problems, and the algorithms in the new models.

Another potential area for understanding the planner behavior is through the IPC problem generators; these could be used to test specific hypotheses about the dependencies we notice. We are also converting non-competition problems<sup>3</sup>, which we hope will extend the benchmarks to other realistic domains.

Our portfolio strategies need to be augmented to reflect the current literature on portfolio learning. For example, it seems reasonable to consider that we could extend the models into an on-line paradigm.

Finally, this work is based in the classical planning paradigm; recent extensions to PDDL relax classical assumptions (Hoffmann & Edelkamp 2005). We hope to fully develop a principled methodology for portfolio construction and then extend it to the newer Temporal and Probabilistic tracks of the IPC.

**Acknowledgments** The author would like to thank his advisor Adele Howe, the students in the AI group at CSU, as well as all the authors of the planning systems.

## References

- Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: An overview. *JAIR* 24:519–579.
- Hoffmann, J. 2004. *Utilizing Problem Structure in Planning: A local Search Approach*. Berlin, New York: Springer-Verlag.
- Howe, A.; Dahlman, E.; Hansen, C.; vonMayrhauser, A.; and Scheetz, M. 1999. Exploiting competitive planner performance. In *Proc. of ECP-99*.
- Roberts, M., and Howe, A. 2006. Directing a portfolio with learning. In Ruml, W., and Hutter, F., eds., *AAAI Workshop on Learning for Search*.
- Roberts, M. 2006. Exploiting portfolio strategy to explore the interaction of problems and algorithms in AI planning. In *Proc. of ICAPS-06, Doctoral Consortium*.
- Witten, I. H., and Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann, 2nd edition.

<sup>3</sup>An undergraduate student, Christina Williams, is working on expanding the problem set.