Search Strategies for Hybrid Search Spaces

Carla Gomes and Bart Selman
Dept. of Computer Science
Cornell University
Ithaca, NY 14850
{gomes,selman}@cs.cornell.edu

Abstract

Recently, there has been much interest in enhancing purely combinatorial formalisms with numerical information. For example, planning formalisms can be enriched by taking resource constraints and probabilistic information into account. The Mixed Integer Programming (MIP) paradigm from Operations Research provides a natural tool for solving optimization problems that combine such numeric and non-numeric information. The MIP approach relies heavily on linear program relaxations and branch-and-bound search. This is in contrast with depth-first or iterative deepening strategies generally used in AI. We provide a detailed characterization of the structure of the underlying search spaces as explored by these search strategies. Our analysis indicates that the traditional approach of identifying dominating search strategies for a given problem domain is inadequate. We show that much can be gained from combining search strategies for solving hard MIP problems, thereby leveraging the strength of different search strategies regarding both the combinatorial and numeric components of the problem.

1. Introduction

Integrating numerical information into standard AI formalism is becoming of increasing importance. For example, in planning, one would like to incorporate resource constraints or a measure of overall plan quality. Other examples are in the area of probabilistic reasoning, where one needs to maximize the probability of the most desirable outcome.

The resulting optimization problems appear to have a novel character in that both the combinatorial aspect of the problem and the optimization part are computationally challenging. More concretely, in the area of planning, we encounter problems for which it is difficult to find a plan that satisfies the hard constraints of the planning task even when ignoring resource constraints. And, of course, finding better plans, with, say, more balanced vehicle loads, is even more difficult. The resulting problems appear to call for a combination of techniques developed for constraint satisfaction to tackle the difficult combinatorics of the problem combined, with techniques from operations research (OR), such as linear program relaxation methods, to handle the numerical optimization aspect of the problem.

A key notion in OR is the use of an *objective function*, with the goal of minimizing or maximizing this function. An objective function is essential in OR models, for two reasons: On one hand, it provides a criterion to guide the search for solutions. Furthermore, it is a way of considering soft constraints. OR experts dealing with real-world applications use the approach of encoding constraints through the objective function, avoiding the use of hard constraints as much as possible. A goal constraint is an objective that is desirable but, if necessary, it can be violated.

In AI, we see more emphasis on feasibility, as opposed to optimization, because problems generally involve a large number of *hard* constraints that are not amenable to being transformed into soft constraints. For example, in a planning formalism, all pre-conditions of an operator generally need to be satisfied in order for the operator to be applicable. We can translate such problems into an OR-style mixed integer programming (MIP) paradigm. The resulting MIP problems have a hard feasibility component, which appears to require new solution strategies. Still this is a promising direction to pursue because the MIP formulation provides a natural way for adding soft constraints, or in general numerical information, to problems. In planning, for example, one can formulate an objective function to minimize the imbalance in vehicle loads ([3], [11], [9], and [13].).

Interestingly, because of the different foci of AI and OR, one on feasibility and the other on optimization, the search paradigms pursued in AI and OR are quite different and specifically taylored towards their respective problem encodings.

OR has traditionally focused on problem formulations where linear program relaxations provide a significant amount of information to prune a branch and bound search. Such methods are less suited for problems, where no good relaxations exist, such as problems with a clear combinatorial component. CSP techniques are more effective at such combinatorial searches. The underlying search paradigm in the CSP approach is a depth first search strategy, enhanced with local propagation and often dynamic variable order strategies. As a result of these different search paradigms, we find that AI search methods tend to branch very quickly, often searching millions of nodes, whereas OR methods explore few nodes but spend a significant amount of time pruning nodes by solving IP relaxations [5].[1]

The challenge we address in this paper is to find a way of synergistically combining AI and OR search methods to solve problem instances with both a difficult feasibility and optimization part. Our focus is on the characterization of problem distributions and runtime profiles. We present a detailed analysis of the properties of these distributions on a range of hard Mixed Integer Programming problems. Based on our findings, we then present a framework for dynamically mixing solution strategies to optimize overall performance. Our experiments are done with a state-of-the-art commercial MIP solver, called CPLEX. The mixed strategies can be directly incorporated into the CPLEX solver.

The paper is structured as follows. In the next section, we review branch-and-bound methods as used for solving MIP problems. In the following sections, we present our data on run time distributions for various search strategies. We then present a general framework for combining these search strategies. We illustrate our approach with an optimization problem (with a hard feasibility component) from the logistics planning domain as well as problems from the MIPLIB library [2]. We consider both the issue of feasibility and optimality of solutions.

2. Branch-and-Bound for MIP

The standard approach in OR for solving MIP problems is to use a branch-and-bound search. First, a linear program (LP) relaxation of the problem instance is considered. In such a relaxation, all variables of the problem are treated as continuous variables. If the solution to the LP relaxation problem has non-integer values for some of the integer variables, we have to branch on one of those variables. This way we create two new subproblems (nodes of the search tree), one with the floor of the fractional value and one with the ceiling. (For the case of binary (0/1) variables, we create an instance with the variable set to 0 and another with the variable set to 1.) The standard heuristic for deciding which variable to branch on is based on the degree of infeasibility of variables ("max infeasibility variable selection"). That is, we select the variable whose non-integer part in the solution of the LP relaxation is closest to 0.5. Informally, we pick the variable whose value is least "decided".

Following the strategy of repeatedly fixing integer variables to integer values will lead at some point to a subproblem with an overall integer solution (provided we are dealing with a feasible problem instance). (Note we call any solution where all the integer variables have integer values an "integer solution".) In practice, it often happens that the solution of the LP relaxation of a subproblem already is an integer solution, in which case we do not have to branch further from this node.

Once we have found an integer solution, its objective function value can be used to prune other nodes in the tree, whose relaxations have worse values. This is because the LP relaxation bounds the optimal solution of the problem. For example, for a minimization problem, the LP relaxation of a node provides a lower-bound on the best possible integer solution.

Another critical issue that determines the performance of branch-and-bound is the way in which the next node to expand is selected. The standard approach, in OR, is to go with a best-bound selection strategy. That is, from the list of nodes (subproblems) to be considered, we select the one with the best LP bound. (This approach is analogous to an $A^\star$ style search. The LP relaxation provides an admissible search heuristic.)

The best-bound node selection strategy is particularly well-suited for reaching an optimal solution (because of the greedy guidance), which has been the traditional focus of much of the research in OR. One significant drawback of this approach is that it may take a long time before the procedure finds an integer solution, because of the breadth first flavor of the search. Also, the approach has serious memory requirements because the full fringe of the tree has to be stored.

Given problems that have a difficult feasibility part, the best-bound approach may take too long before reaching an integer solution. (Note that an integer solution is required before any nodes can be pruned.) In our experiments, we therefore also considered a depth-first node selection strategy. Such a strategy often quickly reaches an integer solution, but may take longer to produce an overall optimal value.

3. Characterizing Search Spaces

As discussed in the introduction, we are interested in problems that combine both a hard feasibility aspect with

[1]One may wonder why the OR approaches do not work well on purely combinatorial problems. The reason is that the LP relaxation may provide little or no information to guide the search. A concrete example are attempts to solve the propositional satisfiability using IP/LP techniques. The problem is that in the standard IP formulations the LP relaxation sets all 0/1 variables to 0.5.

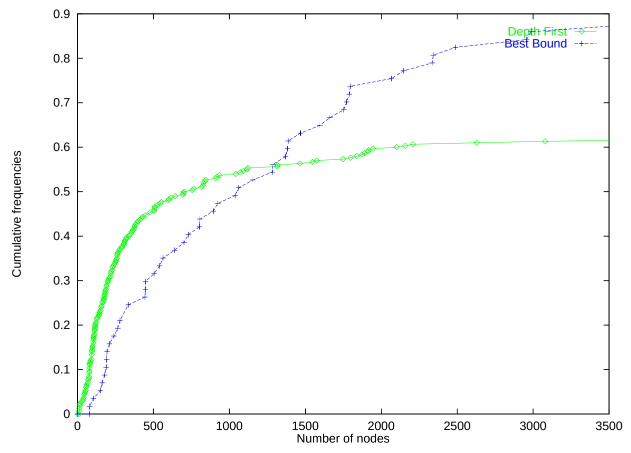

Figure 1. Comparison of runtime profiles for depth-first and best-bound search strategies on a logistics planning problem.

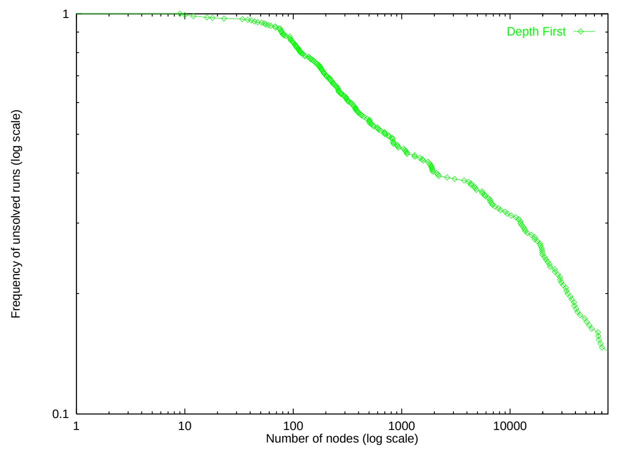

Figure 2. Heavy-tailed behavior of depth-first search.

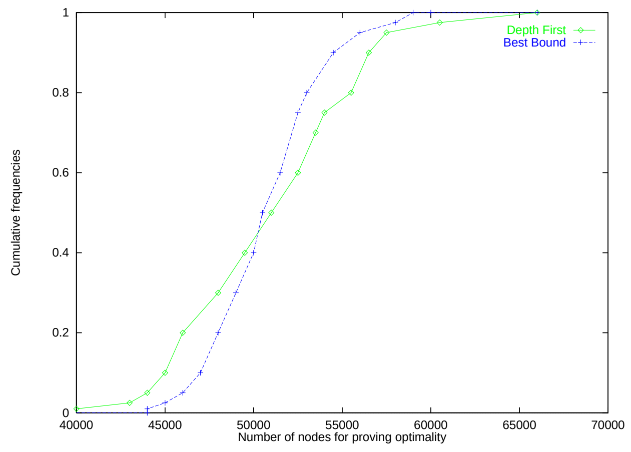

Figure 3. Comparison of runtime profiles for proving optimality for `misc07` from MIPLIB.

a hard optimality component. We obtained examples based on logistics planning problems, formulated as MIP problems. These formulations extend the traditional AI planning approach by combining the hard constraints of the planning operators, initial state, and goal state, with a series of soft contraints capturing resource utilization. For example, one can require that trucks are loaded as close as possible to their maximum capacity. Some initial experimentation with the CPLEX MIP solver showed that these problem instances have indeed a non-trivial feasibility as well as a non-trivial optimization part.[2]

In addition to the logistics problems, we also considered the problems in the MIPLIB problem library [2]. This is a set of 59 benchmark problems, widely used within the OR community. We ran extensive experiments on these problems to look for interesting search behavior. However, we found that the vast majority of the problems have a non-interesting feasibility part. (A feasible integer solution can often be found in less than 1 second.) These problems therefore are mainly of interest for proving optimality of solutions. We include below data on proving optimality of a prototypical problem instance from the library. The instance is called `misc07`; it has 259 0/1 variables, 1 continuous variable, and 212 linear inequality constraints.

In our experiments, we used a state-of-the-art MIP programming package, called CPLEX. CPLEX provides a set of libraries that allows one to customize the branch-and-bound search strategy. For example, one can vary node selection, variable selection, variable setting strategies, the LP solver, etc. We used the default settings for the LP solver, which is for the first node primal-simplex and for subsequent nodes dual-simplex. We modified the search strategies to include some level of randomization.

Randomized search strategies have been shown to to be more robust than deterministic ones when dealing with a variety of problem instances ([1], [12], and [8]). We randomized the variable selection strategy by introducing noise in the ranking of the variables, based on maximum infeasibility. Note that the completeness of the search method is maintained. This is in contrast to the situation for local search. We have experimented with several other randomization strategies. For example, in CPLEX one can assign an apriori variable ranking, which is fixed throughout branch-and-bound. We experimented by randomizing this apriori ranking. We found, however, that the dynamic ran-

[2]We thank Henry Kautz and Joachim Walser for providing us with MIP formulations of the logistic planning problems.

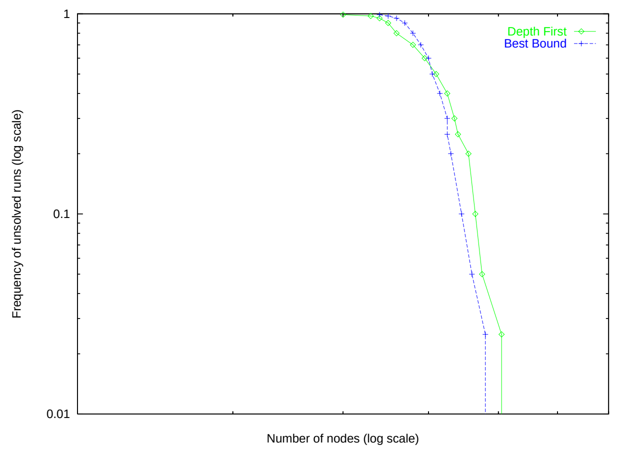

Figure 4. No heavy-tailed behavior for proving optimality.

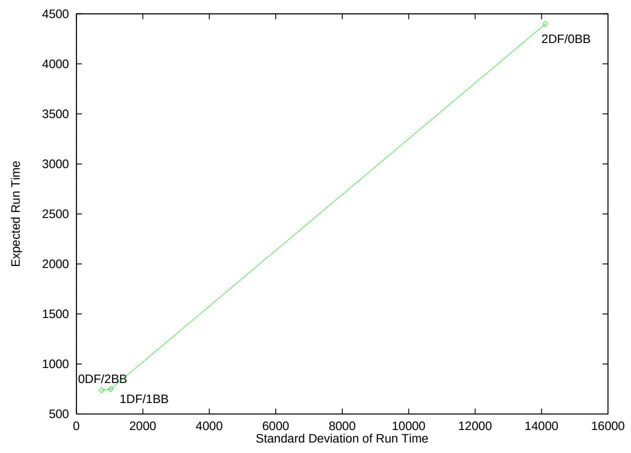

Figure 5. Strategy when using two processors (or by interleaving to two processes) for logistics planning.

domized variable selection strategy, as described above, is more effective.

In figure 1, we compare the runtime profile of a depth-first strategy with a best-bound strategy. The search is terminated when an optimal or near-optimal ($<$10% from optimal) solution is found, but without the requirement of proving optimality.[3] The figure shows the cumulative distribution of solution time (in number of expanded nodes). For example, within 500 nodes, the depth-first search finds a solution on approximately 40% of the runs. Each run had a time limit of 5000 seconds. As we see from the figure, the depth-first search initially outperforms the best-bound search. However, after more than 1500 node expansions, the best-bound becomes more effective. For example, best-bound finds a solution on approximately 75% of the runs with 2000 node expansions or less. In contrast, depth-first search can only find solutions on 55% of the runs with the same number of node expansions. This data is consistent with the observation above that best-bound may take some time to find an intial integer solution. However, once such a solution is found, optimization becomes more effective.

We now consider the runtime distributions more closely. Figure 2 gives a log-log plot of the the complement of the cumulative distribution for the depth-first procedure. For example, from this plot, we see that after 10,000 nodes, approximately 30% of the runs have not yet found the solution. The figures shows a near linear behavior over several orders of magnitude. This is an indication of so-called heavy-tailed behavior which often characterizes complete search methods [7]. In a sense, the time till solution behaves in a very erratic manner: very long runs occur much more frequently than one might expect. Best-bound also appears to exhibit heavy-tailed behavior, but less dramaticly than that for depth-first search.

This erratic search behavior is related to the feasibility part of the search. For example, when we move to proving optimality, the cost distributions are no longer heavy-tailed. This can be seen from figures 3 and 4. These figures show the distributions for proving optimality of the `misc07` problem from the MIPLIB library. In comparison to the previous figures, it is apparent that the runtime for proving optimality exhibits much less variance both for best-bound and for depth-first. (The sharp drop-off in the log-log plot in figure 4 is a clear indication of the absence of heavy-tails.) Our experiments for proving optimality on other instances from the MIPLIB library also did not reveal heavy-tails. We conjecture that indeed proving optimality does not produce heavy-tails since the entire search space has to be explored. This is consistent with the results by Frost *et al.* [4] on constraint satisfaction problems. They show that standard (not heavy-tailed) distributions, such as the Weibull and log-normal distribution, underly the cost of proving inconsistency.

4. Dynamic Strategies

In the previous section we have shown that there are several interesting trade-offs between depth-first branch-and-bound versus best-bound branch-and-bound. In particular, depth-first search performs better early on in the search, whereas best-bound is better on longer runs. We also demonstrated large variations in search cost when look-

[3]One should be careful to distinguish between finding an optimal integer solution and proving that this is indeed *the* optimal solution. Our interest lies in problems where the proof of optimality can be beyond reach of any procedure; however, we can often still find good quality solution.

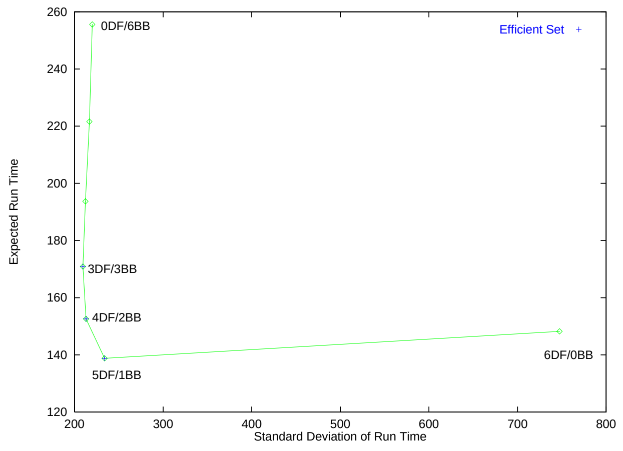

Figure 6. Strategy when using six processors (or running interleaved) for logistics planning problem.

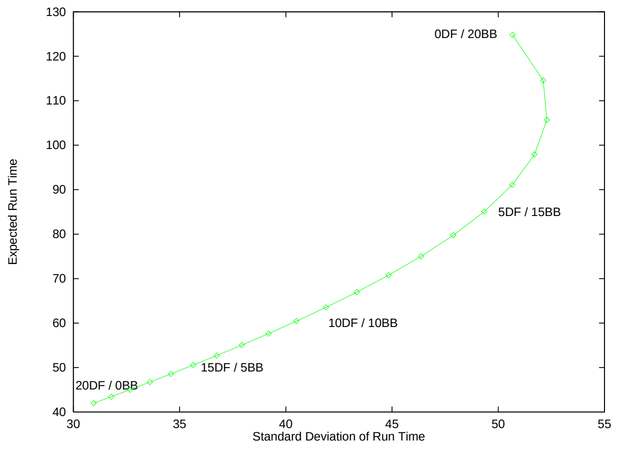

Figure 7. Strategy when using twenty processors (or running interleaved) for logistics planning problem.

ing for a “good” feasible solution.

These results indicate that choosing a single search strategy — which is the standard current practice — may not be the most effective approach when dealing with these types of MIP problems. In fact, as we will show below, one can obtain better results by combining strategies, either by interleaving them or by running them in parallel.

Our analysis extends the portfolio framework, as first studied by Huberman *et al.* [10] in the context of graph coloring methods, and by Gomes and Selman [6] for CSP approaches. In this section, we show the applicability of this framework for the general paradigm of mixed integer programming.

In figure 5, we consider the case of using two processors to solve our feasibility problem for the logistics domain (the same instance as the one considered in figures 1 and 2). The plot gives the expected run time and standard deviation for different ways of combining a branch-and-bound search procedure using depth-first search and best-bound search, assuming two processors. From this plot we see that the best choice, in terms of minimizing expected running time and standard deviation, corresponds to running branch-and-bound with best-bound on both processors. The expected run time of such a strategy is approximately 700 nodes with a standard deviation of around 750. Contrast these values, for example, with the much higher values corresponding to the strategy of running branch-and-bound with depth-first on both processors (average 4397 and standard deviation of 14112).

Figures 6, 7, and 8 show how the mixing strategies change as we increase the number of processors or the amount of interleaving. For example, for the case of six processors, the best strategies are 3DF/3BB, 4DF/2BB, and 5DF/1BB. (We use the notation xDF/yBB to mean running x depth-first processes and y best-bound processes.) These strategies give both a low expected run time and a low standard deviations. There is no clear dominant strategy among those three. In this set, one has to trade a decrease in expected run time for an increase in variance. It is interesting to observe that this analysis shows that in the case of 20 processes, the best strategy corresponds to only using depth-first search (*i.e.*, 20DF/0BB). Figure 8 shows how these profiles change depending on the number of processors.

Finally, figure 9 shows the various profiles for proving optimality for the `misc07` problem instance. The main aspect to note is that in this case the shape of the profiles does not change as the number of processors increases. This is due to the fact that the underlying cost distributions show much less variation. (They are not heavy-tailed.) Nevetherless, mixing strategies is still advantageous, depending on whether one wants to minimize expected cost or variance.

5. Conclusions

We have studied search methods for solving mixed integer programming problems. We focused on problems that include both a hard feasibility part and a relatively difficult optimization component. Such problems arise in AI application when one adds numerical information or soft constraints to a problem with a hard combinatorial component (such as a planning problem). We have shown that there is not one dominant strategy for finding optimal or near-optimal solutions. In fact, depth-first search can effectively complement best-bound strategies, as used in a branch-and-bound approach. We have also presented a framework for

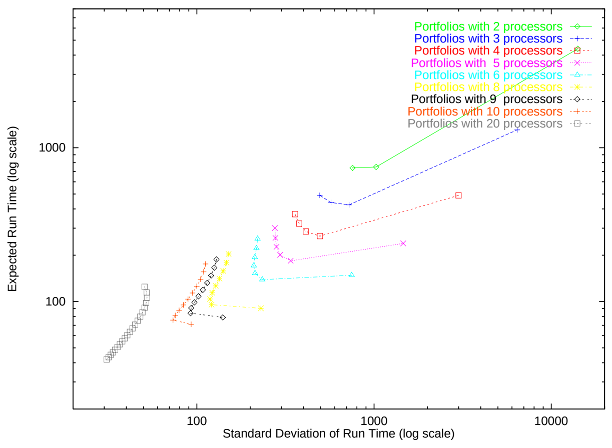

Figure 8. Combined plot of various strategies on logistics planning problem.

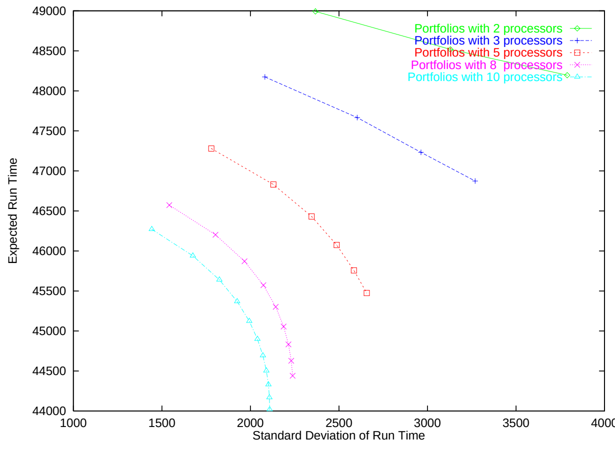

Figure 9. Combined plot of various strategies for proving optimality of `misc07`.

combining different search strategies, taking into consideration the tradeoffs between expected run time and overall variance. Our results show that, by interleaving processes, one can reduce variance and expected run time. Given the wide use of MIP formulations, we hope that our results will have an impact in the area.

Acknowledgments

We would like to thank the reviewers for their comments. The first author is funded by the Air Force Research Laboratory (Rome Research Laboratory) and the Air Force Office of Scientific Research, under the New World Vistas Initiative.

References

[1] R. Bayardo and R.Schrag. Using csp look-back techniques to solve real-world sat instances. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 203–208, New Providence, RI, 1997. AAAI Press.

[2] R. Bixby, C. Ceria, C. McZeal, and M. Savelsberg. An updated mixed integer programming library: Miplib 3.0. *SIAM News*, 1996.

[3] T. Bylander. A linear programming heuristic for optimal planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, New Providence, RI, 1997. AAAI Press.

[4] D. Frost, I. Rish, and L. Vila. Summarizing CSP hardness with continuous probability distributions. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, New Providence, RI, 1997. AAAI Press.

[5] M. Ginsberg. In *Working notes of 1st international AI&OR workshop*, Timberline, OR, USA, 1994.

[6] C. P. Gomes and B. Selman. Algorithm Portfolio Design: Theory vs. Pratice. In *Proceedings of the Thirteenth Conference On Uncertainty in Artificial Intelligence (UAI-97)*, Linz, Austria., 1997. Morgan Kaufman.

[7] C. P. Gomes, B. Selman, and N. Crato. Heavy-tailed Distributions in Combinatorial Search. In *Proceedings of the Third International Conference of Constraint Programming (CP-97)*, Linz, Austria., 1997. Springer-Verlag.

[8] C. P. Gomes, B. Selman, and H. Kautz. Boosting Combinatorial Search Through Randomization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, New Providence, RI, 1998. AAAI Press.

[9] J. Hooker. Modeling-Based Integration of Optimization and Constraint Satsifaction Methods. In *Proceedings of the Workshop on Planning as Combinatorial Search, held in conjunction with AIPS-98*, Pittsburgh, PA, USA, 1998.

[10] B. Huberman, R. Lukose, and T. Hogg. An economics approach to hard computational problems. *Science*, (265):51–54, 1993.

[11] H. Kautz and J. Walser. State-space planning by integer optimization. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, Orlando, FA, 1999. AAAI Press.

[12] M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of las vegas algorithms. *Information Process. Letters*, 1993.

[13] T. Vossen, M. Ball, A. Lotem, and D. Nau. Integer programming models in ai planning: Preliminary experimental results. In *Proceedings of the Workshop on Planning as Combinatorial Search, held in conjunction with AIPS-98*, Pittsburgh, PA, USA, 1998.