Statistical Regimes Across Constrainedness Regions *

Carla P. Gomes (gomes@cs.cornell.edu)
Dpt. of Computer Science, Cornell University, Ithaca, NY

Cèsar Fernández (cesar@eps.udl.es)
Dpt. d'Informàtica, Universitat de Lleida, Jaume II, 69, E-25001 Lleida, Spain

Bart Selman (selman@cs.cornell.edu)
Dpt. of Computer Science, Cornell University, Ithaca, NY

Christian Bessière (bessiere@lirmm.fr)
LIRMM-CNRS, 161 rue Ada, 34392, Montpellier Cedex 5, France

Abstract. Much progress has been made in terms of boosting the effectiveness of backtrack style search methods. In addition, during the last decade, a much better understanding of problem hardness, typical case complexity, and backtrack search behavior has been obtained. One example of a recent insight into backtrack search concerns so-called heavy-tailed behavior in randomized versions of backtrack search. Such heavy-tails explain the large variance in runtime often observed in practice. However, heavy-tailed behavior does certainly not occur on all instances. This has led to a need for a more precise characterization of when heavy-tailedness does and when it does not occur in backtrack search. In this paper, we provide such a characterization. We identify different statistical regimes of the tail of the runtime distributions of randomized backtrack search methods and show how they are correlated with the "sophistication" of the search procedure combined with the inherent hardness of the instances. We also show that the runtime distribution regime is highly correlated with the distribution of the depth of inconsistent subtrees discovered during the search. In particular, we show that an exponential distribution of the depth of inconsistent subtrees combined with a search space that grows exponentially with the depth of the inconsistent subtrees implies heavy-tailed behavior.

Keywords: backtrack search, runtime distributions, heavy-tailed distributions, phase transitions, typical case analysis.

1. Introduction

Significant advances have been made in recent years in the design of search engines for constraint satisfaction problems (CSP), including Boolean satisfiability problems (SAT). For complete solvers, the basic underlying solution strategy is backtrack search enhanced by a series of

* Research supported by the Intelligent Information Systems Institute, Cornell University (AFOSR grant F49620-01-1-0076), MURI (AFOSR grant F49620-01-1-0361) and Ministerio de Educación y Ciencia (TIN-2004 grant 07933-C03-03). We thank the anonymous reviewers for their insightful comments.

increasingly sophisticated techniques, such as non-chronological backtracking [10, 25], fast pruning and propagation methods [14, 24, 22, 28, 4], nogood (or clause) learning (e.g., [7, 26, 2, 21]), and more recently randomization and restarts [12, 13]. For example, in areas such as planning and finite model-checking, we are now able to solve large CSP's with up to a million variables and several million constraints (see e.g., [3, 23]).

The study of problem structure of combinatorial search problems has also provided tremendous insights in our understanding of the interplay between structure, search algorithms, and more generally, typical case complexity. For example, the work on phase transition phenomena in combinatorial search has led to a better characterization of search cost, beyond the worst-case notion of NP-completeness. While the notion of NP-completeness captures the computational cost of the very hardest possible instances of a given problem, in practice, one may not encounter many instances that are quite that hard. In general, CSP problems exhibit an "easy-hard-easy" pattern of search cost, depending on the constrainedness of the problem [15]. The computational hardest instances appear to lie at the phase transition region, the area in which instances change from being almost all solvable to being almost all unsolvable. The discovery of "exceptionally hard instances" reveals an interesting phenomenon : such instances seem to defy the "easy-hard-easy" pattern, they occur in the under-constrained area, but they seem to be considerably harder than other similar instances and even harder than instances from the critically constrained area. This phenomenon was first identified by Hogg and Williams in graph coloring and by Gent and Walsh in satisfiability problems [11, 16]. However, it was shown later that such instances are not inherently difficult; for example, by renaming the variables such instances can often be easily solved [30, 29]. Therefore, the "hardness" of exceptionally hard instances does not reside purely in the instances, but rather in the combination of the instance with the details of the search method. Smith and Grant provide a detailed analysis of the occurrence of exceptionally hard instances for backtrack search, by considering a deterministic backtrack search procedure on sets of instances with the same parameter setting (see e.g., [31]). Recently, researchers have noted that for a proper understanding of search behavior one has to study full runtime distributions[1] [16, 9, 12, 8, 17], as opposed to only considering statistics such as the mean of the distribution. In our work we have focused on the study of randomized backtrack search

[1] Namely the probability (density) function (PDF); the cumulative distribution function (CDF) and the survival function (SF) or tail probability.

algorithms [12]. By studying the runtime distribution produced by a randomized algorithm on a *single* instance, we can analyze the variance caused *solely* by the algorithm, and therefore separate the algorithmic variance from the variance between different instances drawn from an underlying distribution. We have shown previously that the runtime distributions of randomized backtrack search procedures can exhibit extremely large variance, even when solving *the same instance over and over again*, [12, 13]. This work on the study of the runtime distributions of randomized backtrack search algorithms further clarified that the source of extreme variance observed in exceptional hard instances was not due to the inherent hardness of the instances: A randomized version of a search procedure on such instances in general solves the instance easily, even though it has a non-negligible probability of taking very long runs to solve the instance, considerably longer than all the other runs combined. Such extreme fluctuations in the runtime of backtrack search algorithms are nicely captured by so-called heavy-tailed distributions, distributions that are characterized by extremely long tails with some infinite moments [16, 12]. The decay of the tails of heavy-tailed distributions follows a power law — much slower than the decay of standard distributions, such as the normal or the exponential distribution, that have tails that decay exponentially. Further insights into the empirical evidence of heavy-tailed phenomena of randomized backtrack search methods were provided by abstract models of backtrack search that show that, under certain conditions, such procedures provably exhibit heavy-tailed behavior [6, 13, 33, 34].

Main Results

So far, evidence for heavy-tailed behavior of randomized backtrack search procedures on concrete instance models has been largely empirical. Moreover, it is clear that not all problem instances exhibit heavy-tailed behavior. The goal of this work is to provide a better characterization of *when* heavy-tailed behavior occurs, and *when it does not*, when using randomized backtrack search methods. We study the empirical runtime distributions of *randomized* backtrack search procedures across different constrainedness regions of random binary constraint satisfaction models.[2] In order to obtain the most accurate empirical runtime distributions, all our runs are performed without

[2] Hogg and Williams (94) provided the first report of heavy-tailed behavior in the context of backtrack search. They considered a *deterministic* backtrack search procedure on different instances drawn from a given distribution. Our work is of different nature as we study heavy-tailed behavior of the runtime distribution of a given *randomized* backtrack search method on a particular problem instance, thereby isolating the variance in runtime due to different runs of the algorithm.

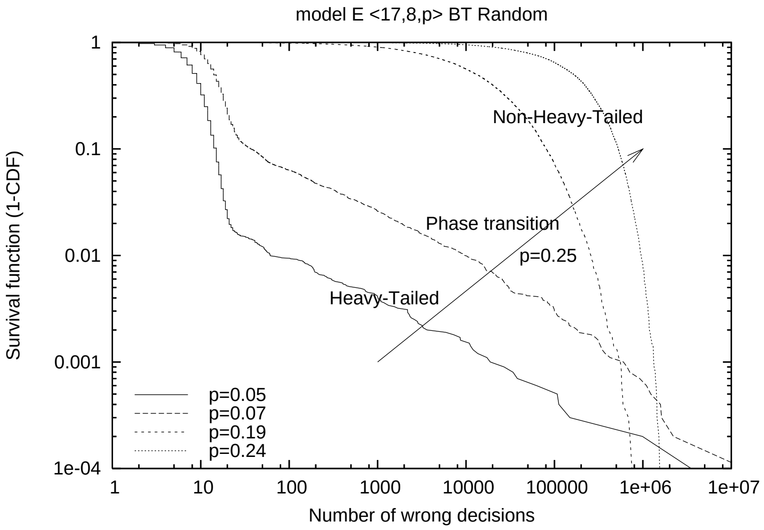


Figure 1. Heavy-tailed (linear behavior) and non-heavy-tailed regime in the runtime of instances of model E $\langle 17, 8, p \rangle$. CDF stands for Cumulative Density Function.

censorship (*i.e.*, we run our algorithms without a cutoff) over the largest possible size. Our study reveals dramatically different statistical regimes for randomized backtrack search algorithms across the different constrainedness regions of the CSP models. Figure 1 provides a preview of our results. The figure plots the runtime distributions (the survival function, *i.e.,* the probability of a run taking more than x backtracks) of a basic randomized backtrack search algorithm (no look-ahead and no look-back), using random variable and value selection, for different constrainedness regions of one of our CSP models (model E; instances with 17 variables and domain size 8). We observe two regions with dramatically different statistical regimes of the runtime distribution.

In the first regime (the bottom two curves in figure 1, $p \leq 0.07$), we see heavy-tailed behavior. This means that the runtime distributions decay slowly. In the log-log plot, we see linear behavior over several orders of magnitude. When we increase the constrainedness of our model (higher p), we encounter a different statistical regime in the runtime distributions, where the heavy-tails disappear. In this region, the instances become inherently hard for the backtrack search algorithm, all the runs become homogeneously long, and therefore the variance of the backtrack search algorithm decreases and the tails of its survival function decay rapidly (see top two curves in figure 1, with $p = 0.19$ and $p = 0.24$; tails decay exponentially).

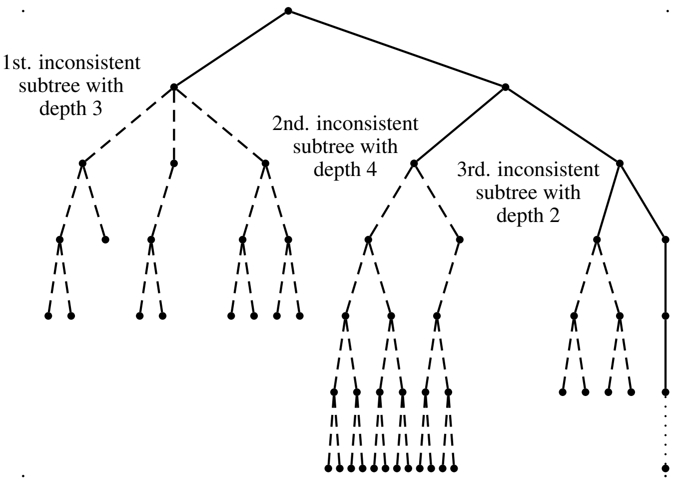


Figure 2. Inconsistent subtrees in backtrack search.

A common intuitive understanding of the extreme variability of backtrack search is that on certain runs the search procedure may hit a very large inconsistent subtree that needs to be fully explored, causing "thrashing" behavior.

To confirm this intuition and in order to get further insights into the statistical behavior of our backtrack search method, we study the inconsistent sub-trees discovered by the algorithm during the search (see figure 2).

The distribution of the depth of inconsistent trees is quite revealing: when the distribution of the depth of the inconsistent trees decreases exponentially (see figure 3, bottom panel, $p = 0.07$) the runtime distribution of the backtrack search method has a power law decay (see figure 3, top panel, $p = 0.07$). In other words, when the backtrack search heuristic has a good probability of finding relatively shallow inconsistent subtrees, and this probability decreases exponentially as the depth of the inconsistent subtrees increases, heavy-tailed behavior occurs. Contrast this behavior with the case in which the survival function of the runtime distribution of the backtrack search method is not heavy-tailed (see figure 3, top panel, $p = 0.24$). In this case, the distribution of the depth of inconsistent trees no longer decreases exponentially (see figure 3, bottom panel, $p = 0.24$).

In essence, these results show that the distribution of inconsistent subproblems encountered during backtrack search is highly correlated with the tail behavior of the runtime distribution. We provide a formal analysis that links the exponential search tree depth distribution with

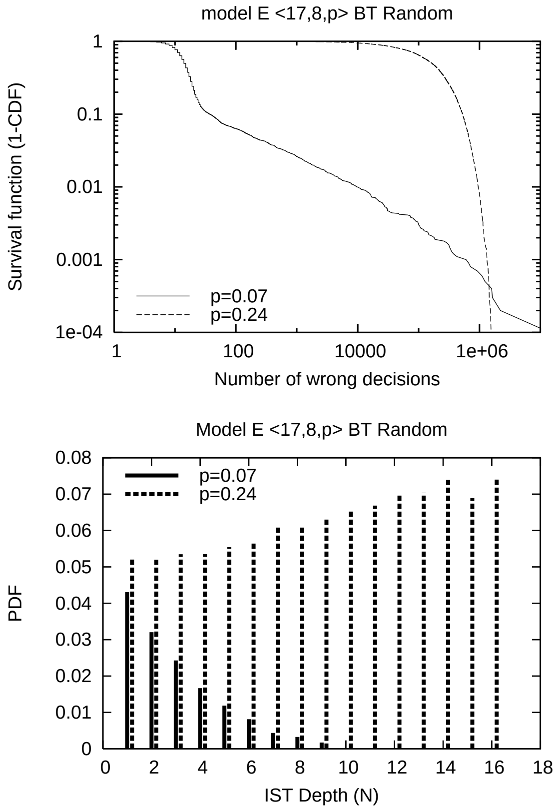


Figure 3. Example of a heavy-tailed instance ($p = 0.07$) and a non-heavy-tailed instance ($p = 0.24$): (top) Survival function of runtime distribution, (bottom) probability density function of depth of inconsistent subtrees encountered during search. The subtree depth for $p = 0.07$ instance is exponentially distributed.

heavy-tailed runtime profiles. As we will see below, the predictions of our model closely match our empirical data.

The structure of the paper is as follows. In the next section we provide definitions of concepts used throughout the paper, namely concepts related to constraint networks and search trees, a description of the random models used for the generation of our problem instances, and a description of the search algorithms that we use in our experimentation. In section 3 we provide empirical results. In section 4 we present a theoretical model of heavy-tailed runtime distributions that considers the distribution of the depth of inconsistent subtrees and the growth of the search space inside such inconsistent subtrees; in this section we also compare the results of our theoretical model with our empirical results. In section 5 we present conclusions and discuss future research directions.

2. Definitions, Problem Instances, and Search Methods

Constraint Networks

A finite binary *constraint network* $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ is defined as a set of n *variables* $\mathcal{X} = \{x_1, \ldots, x_n\}$, a set of *domains* $\mathcal{D} = \{D(x_1), \ldots, D(x_n)\}$, where $D(x_i)$ is the finite set of possible *values* for variable x_i, and a set $\mathcal{C}$ of binary *constraints* between pairs of variables. A constraint C_{ij} on the ordered set of variables (x_i, x_j) is a subset of the Cartesian product $D(x_i) \times D(x_j)$ that specifies the *allowed* combinations of values for the variables x_i and x_j. A *solution* of a constraint network is an instantiation of the variables such that all the constraints are satisfied. The constraint satisfaction problem (CSP) involves finding a solution for the constraint network or proving that none exists. We used a direct CSP encoding and also a Boolean satisfiability encoding (SAT) [32].

Random Problems

The CSP research community has always made a great use of randomly generated constraint satisfaction problems for comparing different search techniques and studying their behavior. Several models for generating these random problems have been proposed over the years. The oldest one, which was the most commonly used until the middle 90's, is *model A*. A network generated by this model is characterized by four parameters $\langle N, D, p1, p2 \rangle$, where N is the number of variables, D the size of the domains, p_1 the probability of having a constraint between two variables, and p_2, the probability that a pair of values is forbidden in a constraint. Notice that the variance in the type of problems generated with the same four parameters can be large, since the actual number of constraints for two problems with the same parameters can vary from one problem to another, and the actual number of forbidden tuples for two constraints inside the same problem can also be different. Model B does not have this variance. In *model B*, the four parameters are again $N, D, p1$, and $p2$, where N is the number of variables, and D the size of the domains. But now, p_1 is the proportion of binary constraints that are in the network (*i.e.*, there are exactly $c = \lfloor p_1 \cdot N \cdot (N-1)/2 \rfloor$ constraints), and p_2 is the proportion of forbidden tuples in a constraint (*i.e.*, there are exactly $t = \lfloor p_2 \cdot D^2 \rfloor$ forbidden tuples in each constraint). Problem classes in this model are denoted by $\langle N, D, c, t\rangle$. In [1] it was shown that model B (and model A as well) can be "flawed" when we increase N. Indeed, when N goes to infinity, we will almost surely have a *flawed* variable (that is, one variable which has all its values inconsistent with one of the constraints involving it). *Model E* was proposed to overcome this weakness. It is a three parameter model, $\langle N, D, p\rangle$, where N and D are the same as in the other models, and $\lfloor p \cdot D^2 \cdot N \cdot (N-1)/2\rfloor$ forbidden pairs of values are selected with repetition out of the $D^2 \cdot N \cdot (N-1)/2$ possible pairs. There is also a way of tackling the problem of flawed variables in model B. In [35] it is shown that by enforcing certain constraints on the relative values of N, D, p_1, and p_2, one can guarantee that model B is sound and scalable, for a range of values of the parameters. In our work, we only considered instances of model B that fall within such a range of values.

Search Trees

A *search tree* is composed of *nodes* and *arcs*. A node u represents an ordered partial instantiation $I(u) = (x_{i_1} = v_{i_1}, \ldots, x_{i_k} = v_{i_k})$. A search tree is rooted at the particular node u_0 with $I(u_0) = \emptyset$. There is an arc from a node u to a node u_c if $I(u_c) = (I(u), x = v)$, x and v being a variable and one of its values. The node u_c is called a child of u and u a parent of u_c. Every node u in a tree T defines a *subtree* T_u that consists of all the nodes and arcs below u in T. The *depth* of a subtree T_u is the length of the longest path from u to any other node in T_u. An inconsistent subtree (IST) is a maximal subtree that does not contain any node u such that $I(u)$ is a solution. (See figure 2.) The maximum depth of an inconsistent subtree is referred to the "inconsistent subtree depth" (ISTD). We denote by $T(A, P)$ the search tree of a backtrack search algorithm A solving a particular instance P, which contains a node for each instantiation visited by A until a solution is reached or inconsistency of P is proved. Once assigned a partial instantiation $I(u) = (x_{i_1} = v_{i_1}, \ldots, x_{i_k} = v_{i_k})$ for node u, the algorithm will search for a partial instantiation of some of its children. In the case that there exists no instantiation which does not violate the constraints, algorithm A will take another value for variable x_{i_k}, and start again checking the children of this new node. In this situation, it is said that a *backtrack* happens. We use the number of wrong decisions or backtracks to measure the *search cost* of a given algorithm [5].[3]

[3] In the rest of the paper sometimes we refer to the search cost as runtime. Even though there are some discrepancies between runtime and the search cost measured in number of wrong decisions or backtracks, such differences are not significant in terms of the tail regime of the distributions.

Algorithms

We studied different search procedures, that differ in the amount of propagation they perform, and in the order in which they generate instantiations. We used three levels of propagation: no propagation (backtracking, `BT`), removal of values directly inconsistent with the last instantiation performed (forward-checking, `FC`), and arc consistency propagation (maintaining arc consistency, `MAC`). We used three different heuristics for variable selection: random selection of the next variable to instantiate (`random`), variables pre-ordered by decreasing degree in the constraint graph (`deg`), and selection of the variable with smallest domain first, ties broken by decreasing degree (`dom+deg`) and always random value selection. For the SAT encodings we used the Davis-Putnam-Logemann-Loveland procedure. More specifically we used a simplified version of Satz [20], without its standard heuristic, and with static variable ordering, injecting some randomness in the value selection heuristics.

Heavy-tailed or Pareto-like Distributions

As we discussed earlier, the runtime distributions of backtrack search methods are often characterized by very long tails or *heavy-tails* (HT). These are distributions that have so-called Pareto like decay of the tails. For a general Pareto distribution $F(x)$, the probability that a random variable is larger than a given value x, *i.e.*, its survival function, is:

$$1 - F(x) = P[X > x] \sim Cx^{-\alpha},\ x > 0,$$

where $\alpha > 0$ and $C > 0$ are constants. I.e., we have power-law decay in the tail of the distribution. These distributions have infinite variance when $1 < \alpha < 2$ and infinite mean and variance when $0 < \alpha \leq 1$. The log-log plot of the tail of the survival function $(1-F(x))$ of a Pareto-like distribution shows linear behavior with slope determined by α.

3. Empirical Results

In the previous section we defined our models and algorithms, as well as the concepts that are central in our study: the runtime distributions of our backtrack search methods and the associated distributions of the depth of the inconsistent subtrees found by the backtrack method. As we discussed in the introduction, our key findings are: (1) we observe different regimes in the behavior of these distributions as we move along different instance constrainedness regions; (2) when the depth of the

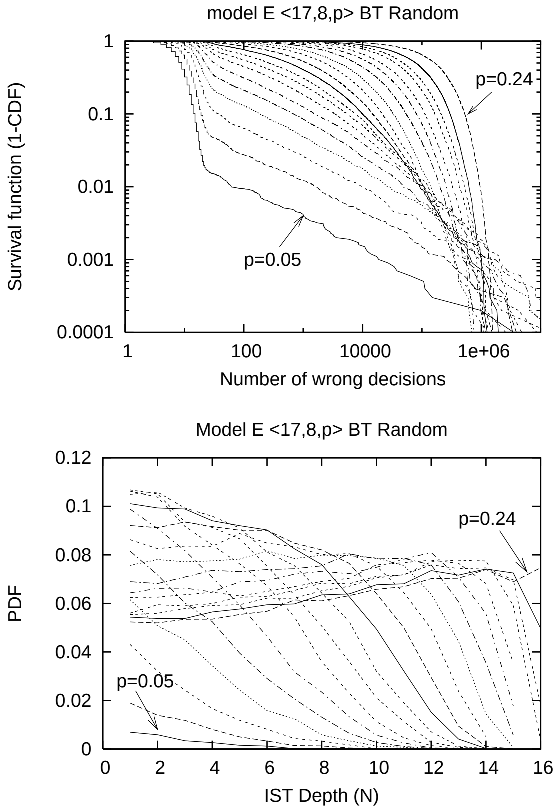


Figure 4. The progression from heavy-tailed regime to non-heavy-tailed regime: (top) survival function of runtime distribution; (bottom) probability density function of the corresponding inconsistent sub-tree depth (ISTD) distribution for BT Random algorithm on instances of model E $\langle 17, 8, p\rangle$. (The value of p increases by 0.01 between curves.)

inconsistent subtrees encountered during the search by the backtrack search method follows an exponential distribution, the corresponding backtrack search method search exhibits heavy-tailed behavior. In this section, we provide the empirical data upon which these findings are based.[4]

We present results for the survival functions of the search cost (number of wrong decisions or number of backtracks) of our backtrack search algorithms. All the plots were computed with at least 10,000 independent executions of a randomized backtrack search procedure on a given (uniquely generated) problem satisfiable instance. For each

[4] Instance generator and data available from the authors.

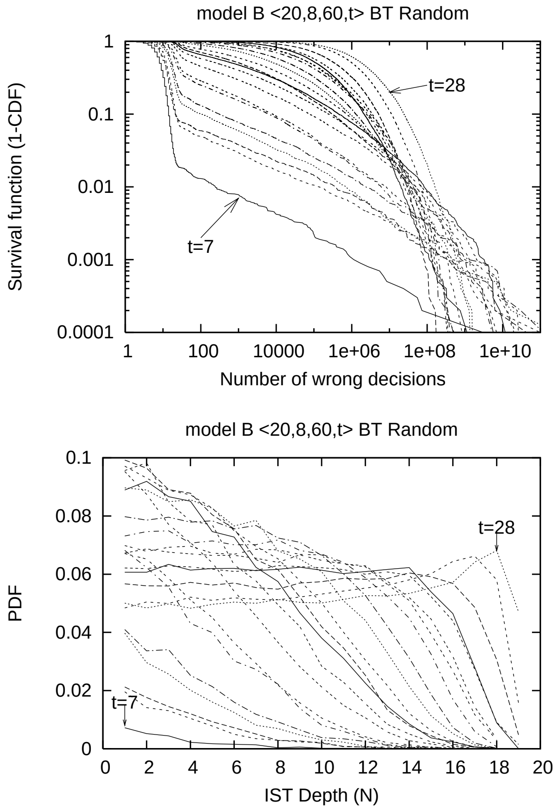


Figure 5. The progression from heavy-tailed regime to non-heavy-tailed regime for BT Random algorithm on instances of model B $\langle 20, 8, 60, t\rangle$. (The value of t increases by 1 between curves.)

parameter setting we considered over 20 instances. In order to obtain more accurate empirical runtime distributions, all our runs were performed without censorship, *i.e.*, we run our algorithms without any cutoff.[5] We also instrumented the code to obtain the information for the corresponding inconsistency sub-tree depth (ISTD) distributions.

Figure 4 (top) provides a detailed view of the heavy-tailed and non-heavy-tailed regions, as well as the progression from one region to the other. The figure displays the survival function (log-log scale) for running (pure) backtrack search with random variable and value selection on instances of Model E with 17 variables and a domain size of 8 for values of p (the constrainedness of the instances) ranging from

[5] For our data analysis, we needed purely uncensored data. We could therefore only consider relatively small problem instances. The results appear to generalize to larger instances.

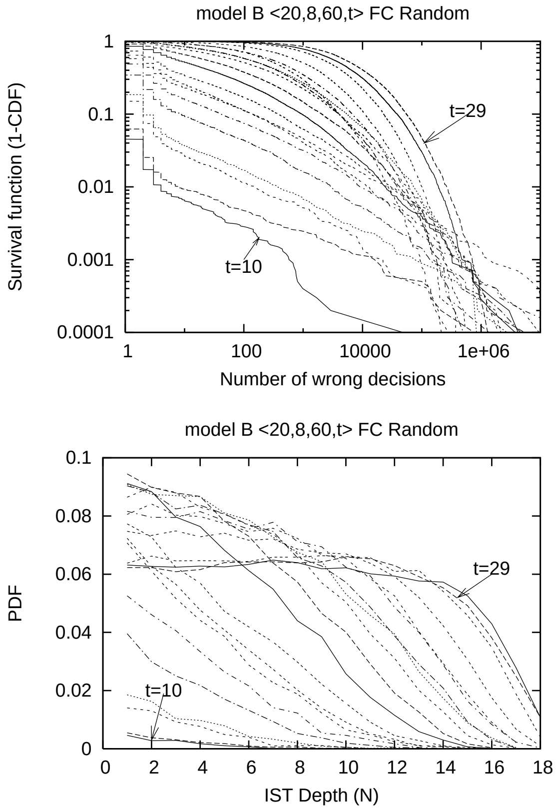


Figure 6. The progression from heavy-tailed regime to non-heavy-tailed regime for FC Random algorithm on instances of model B $\langle 20, 8, 60, t\rangle$. (The value of t increases by 1 between curves.)

$0.05 \leq p \leq 0.24$. We clearly identify the heavy-tailed region in which the log-log plot of the survival functions exhibits linear behavior, while in the non-heavy-tailed region the drop of the survival function is much faster than linear. The transition between regimes occurs around a constrainedness level of $p = 0.09$.

Interestingly, we observe that some of the curves that correspond to the heavy-tailed regime cross the curves corresponding to the non-heavy-tailed regime. That is, some of the longest runs for the heavy-tailed instances are longer than the longest runs for the non-heavy-tailed instances. The intuitive explanation for this is that in the heavy-tailed regime, the backtrack search will now and then encounter an inconsistent subproblem that is quite under-constrained. The search trees of such sub-problems are actually larger than any inconsistent sub-

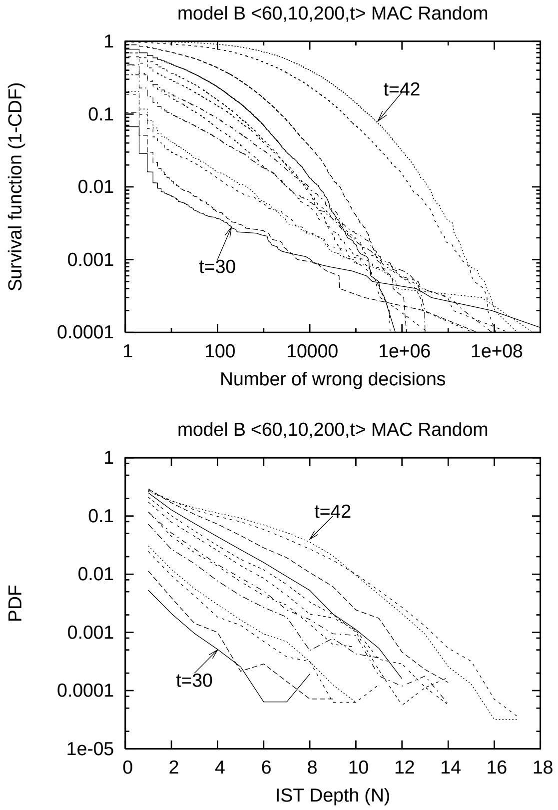


Figure 7. The progression from heavy-tailed regime to non-heavy-tailed regime for MAC Random algorithm on instances of model B $\langle 60, 10, 200, t\rangle$. (The value of t increases by 1 between curves.)

problems encountered in the non-heavy-tailed regime, where instances are more constrained and therefore more pruning occurs during search.

Figure 4 (bottom) depicts the probability density function of the corresponding inconsistent sub-tree depth (ISTD) distributions. The figure shows that while the ISTD distributions that correspond to the heavy-tailed region have an exponential behavior (see section 4 where we compare the predictions of our theoretical model against our data, for a heavy-tailed instance of model E $\langle 17, 8, 0.07\rangle$), the ISTD distributions that correspond to the non-heavy-tailed region are quite different from the exponential distribution.

For all the backtrack search variants that we considered on instances of model E, including the DPLL procedure, we observed a pattern similar to that of figure 4. (See bottom panel of figure 9 for the DPLL data.)

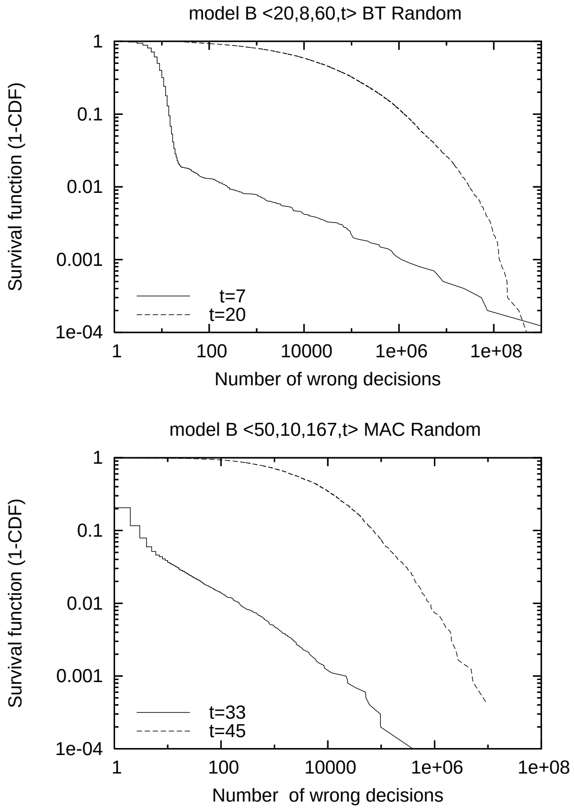


Figure 8. Heavy-tailed and non-heavy-tailed regimes for instances of model B: (top) $\langle 20, 8, 60, t\rangle$, using `BT-random`, (bottom) $\langle 50, 10, 167, t\rangle$, using `MAC-random`.

We also observed a similar behavior — a transition from heavy-tailed region to non-heavy-tailed region with increased constrainedness — for instances of Model B, for different problem sizes and different search strategies. Figures 5, 6 and 7 show the same qualitative behavior as in figure 4, for instances of model B, with search strategy `BT-random`, `FC-random`, and `MAC-random`, respectively. These figures were generated by running the corresponding randomized backtrack search procedure on the same instance (10,000 runs per instance). Again, we clearly see the progression from the heavy-tailed to the non-heavy-tailed region. Note that for the sake of clarity, for `MAC-random` (figure 7), the probability density function is plotted in log scale (Y-axis) and therefore the exponential distributions exhibit linear behavior. Figure 8 (top) shows the survival functions of runtime distributions of instances of model B $\langle 20, 8, 60, t\rangle$, for different levels of constrainedness, solved with `BT-random`. Figure 8 (bottom) shows the survival functions of runtime

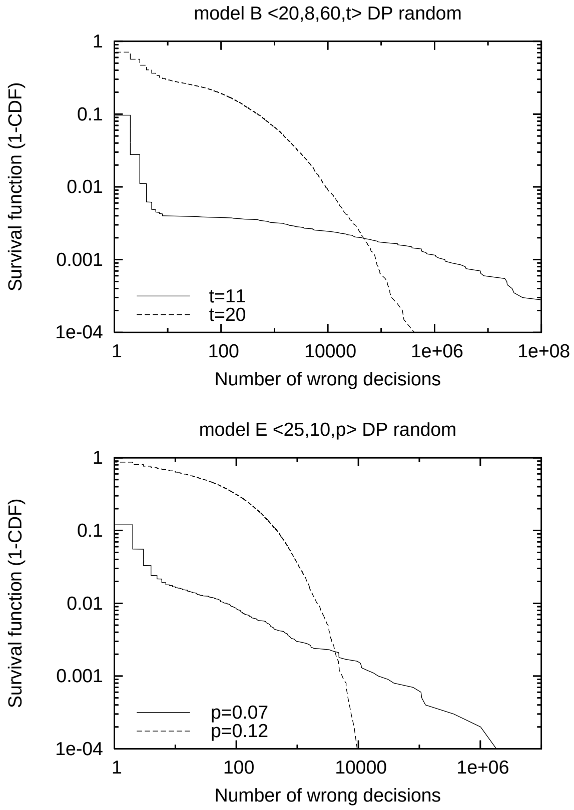


Figure 9. Heavy-tailed and non-heavy-tailed regimes for instances of (top) model B $\langle 20, 8, 60, t\rangle$, using `DP-random` (DPLL procedure with static variable ordering and random value selection) and (bottom) model E $\langle 25, 10, p\rangle$ using `DP-random`.

distributions of instances of model B $\langle 50, 10, 167, t\rangle$, for different levels of constrainedness, solved with `MAC-random`. The top panel of figure 9 gives the DPLL data for instances of model B. Again, the two different statistical regimes of the runtime distributions are quite clear in these plots.

To summarize our findings:

– For both models (B and E), for CSP and SAT encodings, for the different backtrack search strategies, we clearly observe two different statistical regimes — a heavy-tailed and a non-heavy-tailed regime.

– As constrainedness increases (p increases), we move from the heavy-tailed region to the non-heavy-tailed region.

- The transition point from heavy-tailed to non-heavy-tailed regime is dependent on the particular search procedure adopted. As a general observation, we note that as the efficiency (and, in general, propagation strength) of the search method increases, the extension of the heavy-tailed region increases and therefore the heavy-tailed threshold gets closer to the phase transition.

- Exponentially distributed inconsistent sub-tree depth (ISTD) combined with exponential growth of the search space as the tree depth increases implies heavy-tailed runtime distributions. We observe that as the ISTD distributions move away from the exponential distribution, the runtime distributions become non-heavy-tailed.

These results suggest that heavy-tailed behavior in the cost distributions depends on the efficiency of the search procedure as well as on the level of constrainedness of the problem. Increasing the algorithm efficiency tends to shift the heavy-tail threshold closer to the phase transition. We conjecture that even when considering more sophisticated search methods involving, e.g., intelligent backtracking and no-good learning (see e.g., [7, 26, 2, 21]), one will encounter qualitatively the same pattern, albeit for larger problem instances. Although we have not yet studied this pattern for more sophisticated search strategies, we have identified heavy-tailed behavior for SAT solvers with no-good learning and probing in earlier work on structured problems [13].

For both models, B and E, and for the different search strategies, we also clearly observed that when the ISTD follows an exponential distribution, the corresponding distribution is heavy-tailed. In the next section we present a theoretical model and validate it against our data.

4. Validation

Let X be the search cost of a given backtrack search procedure, $P_{istd}[N]$ be the probability of finding an inconsistent subtree of depth N during search, and $P[X > x|N]$ the probability of having a inconsistent search tree of size larger than x, given a tree of depth N. Assuming that the inconsistent search tree depth follows an exponential distribution in the tail and the search cost inside an inconsistent tree grows exponentially, then the cost distribution of a search method is lower bounded by a Pareto distribution. More formally:

Theoretical Model

Assumptions:

– $P_{istd}[N]$ is exponentially distributed in the tail, i.e.,

$$P_{istd}[N] = B_1 e^{-B_2 N}, N > n_0 \tag{1}$$

where B_1, B_2, and n_0 are constants.

– $P[X > x|N]$ is modeled as a complementary Heavyside function, $1 - H(x - k^N)$, where k is a constant and

$$H(x-a) = \begin{cases} 0, x < a \\ 1, x \geq a \end{cases}$$

Then, $P[X > x]$ is Pareto-like distributed

$$P[X > x] \approx \beta x^{-\alpha}$$

for $x > k^{n_0}$, where α and β are constants.
Derivation of result:

Note that $P[X > x]$ is lower bounded as follows

$$P[X > x] \geq \int_{N=0}^{\infty} P_{istd}[N]\, P[X > x|N] dN \tag{2}$$

This is a lower bound since we consider only one inconsistent tree contributing to the search cost, when in general there are more inconsistent trees. Given the assumptions above, Eq. (2) results

$$\begin{aligned} P[X > x] \;&\geq\; \int_{N=0}^{\infty} P_{istd}[N]\,(1 - H(x - k^N))dN \\ &= \int_{N=\frac{\ln x}{\ln k}}^{\infty} P_{istd}[N] dN \end{aligned} \tag{3}$$

Since $x > k^{n_0}$, we can use Eq.(1) for $P_{istd}[N]$, so Eq.(3) results in:

$$P[X > x] \;\geq\; \int_{N=\frac{\ln x}{\ln k}}^{\infty} B_1 e^{-B_2 N} dN = \frac{B_1}{B_2}\, e^{-B_2 \frac{\ln x}{\ln k}} = \beta x^{-\alpha}$$

with

$$\alpha = \frac{B_2}{\ln k}; \quad \beta = \frac{B_1}{B_2}$$

Note that when $1 < \alpha < 2$, X has infinite variance; when $\alpha \leq 1$, X has infinite mean and infinite variance.

In order to empirically validate our theoretical model we compared its predictions against our data, for instances of model B and E. Figure 10 and figure 11 illustrate the comparison for two instances. Figure 10 concerns an instance from model B $\langle 20, 8, 60, 7\rangle$, running `BT-random`, the same instance plotted in figure 8 (top), for which heavy-tailed behavior was observed ($t = 7$). The plots in figure 10 provide the regression data and fitted curves for the parameters B_1, B_2, and k, using $n_0 = 1$. The good quality of the linear regression fit suggests that our assumptions are very reasonable. Based on the estimated values for k, B_1, and B_2, we then compare the lower bound predicted using the formal analysis presented above with the empirical data. As we can see from figure 10, the theoretical model provides a good (tight) lower bound for the empirical data. In figure 11 we illustrate the comparison between our theoretical model and our data for an instance from model E $\langle 17, 8, 0.07\rangle$, also running `BT-random`, the same instance plotted in figure 3 (top), for which heavy-tailed behavior was observed ($p = 0.07$). As in the previous case, the theoretical model lower bounds the empirical data quite well.

5. Conclusions and Future Work

We have studied the runtime distributions of complete backtrack search methods on instances of well-known random CSP binary models. Our results reveal different regimes in the runtime distributions of the backtrack search procedures and corresponding distributions of the depth of the inconsistent sub-trees. We see a changeover from heavy-tailed behavior to non-heavy-tailed behavior when we increase the constrainedness of the problem instances. The exact point of changeover depends on the sophistication of the search procedure, with more sophisticated solvers exhibiting a wider range of heavy-tailed behavior. In the non-heavy-tailed region, the instances become harder and harder for the backtrack search algorithm, and the runs become nearly homogeneously long. We have also shown that there is a clear correlation between the the distributions of the depth of the inconsistent sub-trees encountered by the backtrack search method and the heavy-tailedness of the runtime distributions, with exponentially distributed sub-tree depths leading to heavy-tailed search. To further validate our findings, we compared our theoretical model, which models exponentially distributed subtrees in the search space, with our empirical data: the theoretical model provides a good (tight) lower bound for the empirical data. Our findings on the distribution of inconsistent subtrees in backtrack search give, in

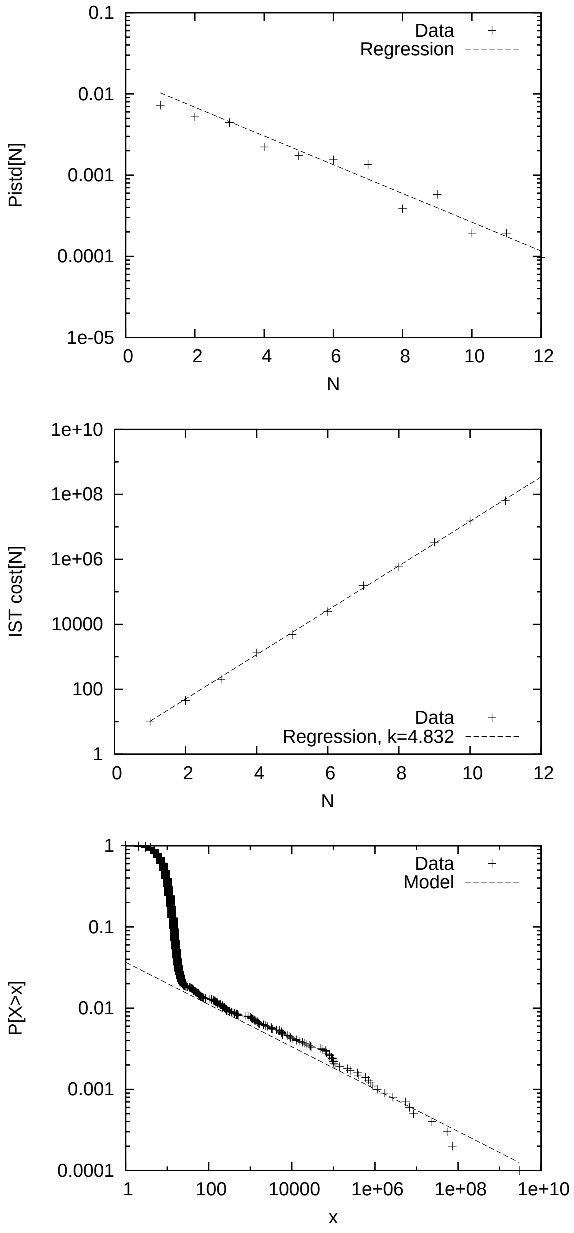
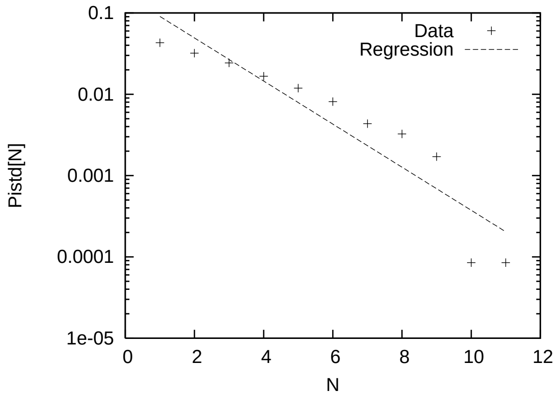
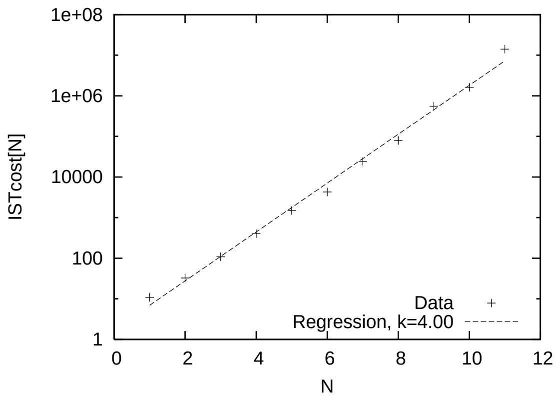


Figure 10. Regressions for the estimation of B1=0.015, B2=0.408 (top plot; quality of fit $R^2 = 0.88$), and $k = 4.832$ (middle plot; $R^2 = 0.98$) and comparison of lower bound based on the theoretical model with empirical data (bottom plot). We have $\alpha = B_2/\ln(k) = 0.26$ from our model; $\alpha = 0.27$ directly from runtime data. Model B $\langle 20, 8, 60, 7\rangle$, using `BT-random`.





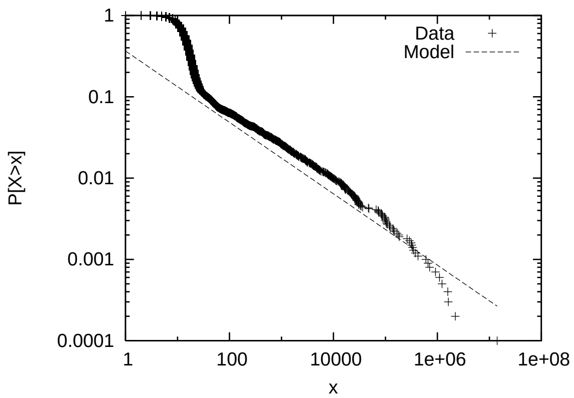


Figure 11. Regressions for the estimation of B1=0.167, B2=0.610 (top plot; quality of fit $R^2 = 0.86$), and $k = 4.00$ (middle plot; $R^2 = 0.99$) and comparison of lower bound based on the theoretical model with empirical data (bottom plot). We have $\alpha = B_2/\ln(k) = 0.439$ from our model; $\alpha = 0.443$ directly from runtime data. Model E $\langle 17, 8, 0.07 \rangle$, using `BT-random`.

effect, information about the inconsistent subproblems that are created during the search.

We believe that these results can be exploited in the design of more efficient restart strategies and backtrack solvers. Note that a restart strategy provably eliminates heavy-tailed behavior [13]. Furthermore, heavy-tailed distributions are characterized by a wide range of runtimes. By using fast restarts not only does a solver avoid the long-tailed runs, it also has a good chance of encountering very short successful runs. Many current state-of-art SAT solvers exploit such restart strategies [23]. However, current restart schemes are somewhat adhoc, and are often tuned by hand. In practice, it has been observed that relatively fast restarts are most effective. One interesting direction for future research is whether one can detect during a single run whether the solver has reached a large inconsistent subtree. For preliminary results on this issue, see [18, 19]. More generally, an important research direction is the design of more sophisticated restart strategies.

Another interesting direction for future research involves the study of more structured problem domains. In recent work, we have introduced the notion of a "backdoor set", which is a special set of variables such that, when assigned values, the polytime propagation mechanism of the solver can resolve the remaining instance [33, 34]. (For a closely related notion, see the work on treewidth and cutset [27, 8].) We have found that practical, structured instances, have surprisingly small sets of backdoor variables (often only a few dozen from among thousands of variables). Moreover, variable selection heuristics are quite good at finding those backdoor variables, which explains the presence of very short, successful runs in structured domains. Further work is needed to better understand the semantics of backdoor sets, namely by relating the presence of small backdoor sets with the structure of different problem domains. In other recent work we have also provided some initial results on the connections between backdoors, restarts, and heavy-tails in combinatorial search [34]. We hope the findings reported in this paper will provide additional insights on the nature of heavy-tailed behavior in combinatorial search and lead to further improvements in the design of restart strategies and search methods.

References

1. Achlioptas, D., L. Kirousis, E. Kranakis, D. Krizanc, M. Molloy, and Y. Stamatiou: 1997, 'Random Constraint Satisfaction: a More Accurate Picture'. In: *Proceedings CP'97*. Linz, Austria, pp. 107–120.
2. Bayardo, R. and D. Miranker: 1996, 'A complexity analysis of space-bounded learning algorithms for the constraint satisfaction problem'. In: *Proceedings*

of the Thirteenth National Conference on Artificial Intelligence (AAAI-96). Portland, OR, pp. 558–562.

3. Berre, D. L. and L. Simon: 2004, ‘Fifty-five solvers in vancouver: The sat 2004 competition.’. In: *Proceedings of SAT’04*.
4. Bessière, C. and J. Régin: 1996, ‘MAC and combined heuristics: two reasons to forsake FC (and CBJ?) on hard problems’. In: *Proceedings CP’96*. Cambridge MA, pp. 61–75.
5. Bessière, C., B. Zanuttini, and C. Fernández: 2004, ‘Measuring Search Trees’. In: B. Hnich (ed.): *Proceedings ECAI’04 Workshop on Modelling and Solving Problems with Constraints*. Valencia, Spain.
6. Chen, H., C. Gomes, and B. Selman: 2001, ‘Formal Models of Heavy-tailed Behavior in Combinatorial Search’. In: *Proceedings CP’01*. Paphos, Cyprus, pp. 408–421.
7. Dechter, R.: 1990, ‘Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition’. *Artificial Intelligence* **41**(3), 273–312.
8. Dechter, R.: 2003, *Constraint Processing*. Morgan Kaufmann.
9. Frost, D., I. Rish, and L. Vila: 1997, ‘Summarizing CSP Hardness with Continuous Probability Distributions’. In: *AAAI-97*. Providence RI, pp. 327–333.
10. Gaschnig, J.: 1977, ‘A General Backtrack Algorithm that Eliminates Most Redundant Tests’. In: *Proceedings IJCAI’77*. Cambridge MA, p. 447.
11. Gent, I. and T. Walsh: 1994, ‘Easy Problems are Sometimes Hard’. *Artificial Intelligence* **70**, 335–345.
12. Gomes, C., B. Selman, and N. Crato: 1997, ‘Heavy-tailed Distributions in Combinatorial Search’. In: *Proceedings CP’97*. Linz, Austria, pp. 121–135.
13. Gomes, C. P., B. Selman, N. Crato, and H. Kautz: 2000, ‘Heavy-tailed phenomena in satisfiability and constraint satisfaction problems’. *J. of Automated Reasoning* **24**(1–2), 67–100.
14. Haralick, R. and G. Elliot: 1980, ‘Increasing Tree Search Efficiency for Constraint Satisfaction Problems’. *Artificial Intelligence* **14**, 263–313.
15. Hogg, T., B. Huberman, and C. Williams: 1996, ‘Phase Transitions and Search Problems’. *Artificial Intelligence* **81 (1-2)**, 1–15.
16. Hogg, T. and C. Williams: 1994, ‘The Hardest Constraint Problems: a Double Phase Transition’. *Artificial Intelligence* **69**, 359–377.
17. Hoos, H. H. and T. Stützle: 2004, *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann.
18. Horvitz, E., Y. Ruan, C. Gomes, H. Kautz, B. Selman, and M. Chickering: 2001, ‘A Bayesian Approach to Tackling Hard Computational Problems’. In: *Proceedings of the Seventeenth Conference On Uncertainty in Artificial Intelligence (UAI-01)*.
19. Kautz, H., E. Horvitz, Y. Ruan, C. Gomes, and B. Selman: 2002, ‘Dynamic Restart Policies’. In: *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*. Edmonton, Canada.
20. Li, C. and Ambulagan: 1997, ‘Heuristics Based on Unit Propagation for Satisfiability Problems’. In: *Proceedings IJCAI’97*. Nagoya, Japan, pp. 366–371.
21. Marques-Silva, J. P. and K. A. Sakallah: 1999, ‘GRASP - A search algorithm for propositional satisfiability’. *IEEE Transactions on Computers* **48**(5), 506–521.
22. Mohr, R. and T. Henderson: 1986, ‘Arc and Path Consistency Revisited’. *Artificial Intelligence* **28**, 225–233.

23. Moskewicz, M., C. Madigan, Y. Zhao, L. Zhang, and S. Malik: 2001, 'Chaff: Engineering an Efficient SAT Solver'. In: *Proceedings of the 39th Design Automation Conference*. Las Vegas.
24. Nadel, B.: 1989, 'Constraint Satisfaction Algorithms'. *Computational Intelligence* **5**, 188–224.
25. Prosser, P.: 1993a, 'Domain Filtering Can Degrade Intelligent Backtrack Search'. In: *Proceedings IJCAI'93*. Chambry, France, pp. 262–267.
26. Prosser, P.: 1993b, 'Hybrid algorithms for the constraint satisfaction problem'. *Computational Intelligence* **9**(3), 268–299.
27. Rish, I. and R. Dechter: 2000, 'Resolution versus Search: Two Strategies for SAT'. *J. of Automated Reasoning* **24**(1/2), 225–275.
28. Sabin, D. and E. Freuder: 1994, 'Contradicting Conventional Wisdom in Constraint Satisfaction'. In: *Proceedings PPCP'94*. Seattle WA.
29. Selman, B. and S. Kirkpatrick: 1996, 'Finite-Size Scaling of the Computational Cost of Systematic Search'. *Artificial Intelligence* **81**(1–2), 273–295.
30. Smith, B. and S. Grant: 1995, 'Sparse Constraint Graphs and Exceptionally Hard Problems'. In: *Proceedings IJCAI'95*. Montral, Canada, pp. 646–651.
31. Smith, B. and S. Grant: 1997, 'Modelling Exceptionally Hard Constraint Satisfaction Problems'. In: *Proceedings CP'97*. Linz, Austria, pp. 182–195.
32. Walsh, T.: 2000, 'SAT vs CSP'. In: *Proceedings CP'00*. Singapore, pp. 441–456.
33. Williams, R., C. Gomes, and B. Selman: 2003a, 'Backdoors to Typical Case Complexity'.
34. Williams, R., C. Gomes, and B. Selman: 2003b, 'On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search'. In: *Proceedings of Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT-03)*.
35. Xu, K. and W. Li: 2000, 'Exact Phase Transition in Random Constraint Satisfaction Problems'. *JAIR* **12**, 93–103.