Pareto-Like Distributions in Random Binary CSP

Christian Bessiere, Cèsar Fernàndez, Carla P. Gomez, Magda Valls

Pareto-like Distributions in Random Binary CSP*

Christian Bessière[1], Cèsar Fernández[2] , Carla P. Gomes[3] , Magda Valls[4]

LIRMM-CNRS [1]
161 rue Ada, 34392
Montpellier Cedex 5, France
bessiere@lirmm.fr

Dept. of Computer Science [2]
Universitat de Lleida
Jaume II 69, Lleida, Spain
cesar@eup.udl.es

Dept. of Computer Science [3]
Cornell University
Ithaca, NY 14853, USA
gomes@cs.cornell.edu

Dept. of Mathematics [4]
Universitat de Lleida
Jaume II 69, Lleida, Spain
magda@eup.udl.es

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 variations in run-time 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. In particular, we will identify different statistical regimes in the parameter space of a standard instance generation model. We show that whether backtrack search is heavy-tailed or not depends on the statistical regime of the instance space.

Keywords: constraint satisfaction problems, heavy-tailed distributions, inconsistent search subtrees.

1 Introduction

In recent years we have made great strides in designing more efficient backtrack search methods for solving constraint satisfaction problems (CSP), including Boolean satisfiability problems (SAT). Current state-of-the-art backtrack solvers use a combination of strong search heuristics, fast pruning and propagation techniques, non-chronological backtracking and nogood learning, and more recently randomization and restarts. For example, in areas such as

*Research supported by AFOSR, grant F49620-01-1-0076 (Intelligent Information Systems Institute) and F49620-01-1-0361 (MURI grant on Cooperative Control of Distributed Autonomous Vehicles in Adversarial Environments), CICYT, TIC2001-1577-C03-03 and DARPA, F30602-00-2-0530 (Controlling Computational Cost: Structure, Phase Transitions and Randomization) and F30602-00-2-0558 (Configuring Wireless Transmission and Decentralized Data Processing for Generic Sensor Networks). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR, DARPA, or the U.S. Government.

planning and finite model-checking, we are now able to solve large CSP's with up to a million variables and five million constraints. 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 that many instances that are quite that hard. We now know that, in general, CSP problems exhibit an "easy-hard-easy" pattern of search cost, depending on the constrainedness of the problem [10, 7]. 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. "Exceptionally hard instances" seem to defy this pattern: such instances occur in the under-constrained area, they are considerably harder than other similar instances and even harder than instances from the critically constrained area. However, different algorithms encounter different "exceptionally hard instances". Therefore, the "hardness" of exceptionally hard instances does not necessarily reside purely in the instances, but rather in the combination of the instance with the details of the search method [4, 12]. The work on the study of run time distributions of backtrack search algorithms further explains this phenomenon — the performance of backtrack search algorithms can exhibit extremely large variance, even on the *same* instance, *just* by introducing a small element of randomness into its heuristic, for example by breaking ties randomly. Such extreme fluctuations in the run time 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 [5, 6]. 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 log-normal, or the exponential distribution, that have tails that decay exponentially. Further insights into the empirical evidence of heavy-tailed phenomena are provided by *abstract* models of backtrack search that show that, under certain conditions, such procedures *provably* exhibit heavy-tailed behavior. In this work *backdoor* variables are a key notion [2, 13]. A set of variables forms a backdoor for a problem instance if there is a value assignment to these variables such that the simplified sub-problem can be solved in polynomial time by the propagation and simplification mechanism of the CSP solver under consideration. Intuitively, the backdoor corresponds to a set of variables, such that when set correctly, the sub-solver can solve the remaining problem easily. A backtrack search algorithm exhibits heavy-tailed behavior when the success probability of the heuristic of the backtrack search method is sufficiently low and the backdoor size is sufficiently small [2, 13].

In this paper we report a different approach. We study the empirical run time distributions of *concrete* backtrack search algorithms across the different constrainedness regions of random binary constraint satisfaction problem models (Model A, B, and E [1, 3]). In order to obtain more accurate empirical run time distributions, all our runs are performed without censorship (*i.e.*, we run our algorithms without a cutoff). 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 run time distributions (the survival function, *i.e.*, the complement to one of the cumulative distribution function), of a simple backtrack search algorithm (no look-ahead and no look-back), using variable random ordering heuristic, with random value selection, for different constrainedness regions of model E (instances with 20 variables and domain

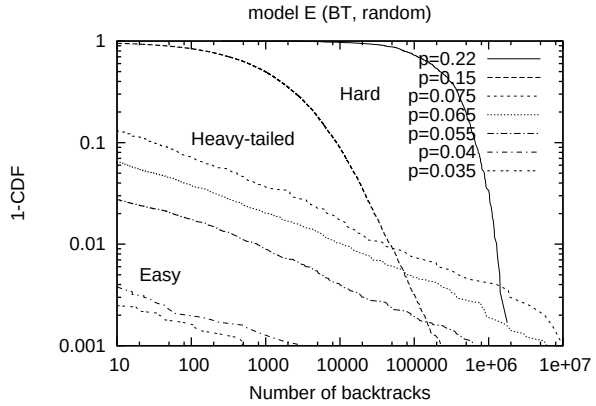

Figure 1: Survival function of the number of backtracks to solve different instances of model E with 20 variables and a domain size of 10. The parameter p captures the constrainedness of the instances. Heavy-tailed regime (curves with linear behavior) and a non-heavy-tailed regime can be identified.

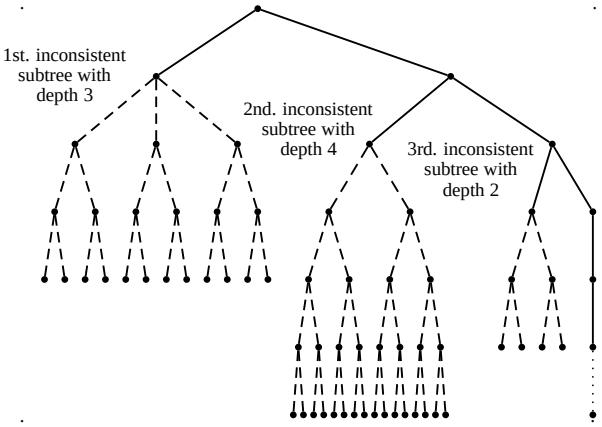

Figure 2: Inconsistent sub-trees in backtrack search.

size 10). Two regions, with dramatically different statistical regimes of the survival function of run time distributions of the backtrack search algorithm can be clearly identified. There is a region in which the tails of the survival distribution decay slowly, exhibiting power law decay. In Figure 1, the curves corresponding to instances with $p \leq 0.075$ exhibit power law decay, which is easily identified by their linear behavior. To some extent the boundary of this region corresponds to a threshold for the backtrack search algorithm. After this region, the instances become too 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 exponentially (see Figure 1, e.g., the curve corresponding to the instance with $p = 0.22$ exhibits exponential decay, much faster than linear behavior).

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, which are associated with the well known phenomenon of *thrashing* in backtrack search (see Figure 2). The distribution of the depth of inconsistent trees is quite revealing: when the run time distribution of the backtrack search method has a power law decay (see Figure 3, left panel, $p = 0.075$), the distribution of the depth of the inconsistent trees decreases exponentially (see Figure 3, right panel, $p = 0.075$). In other words, the backtrack search heuristic has a higher

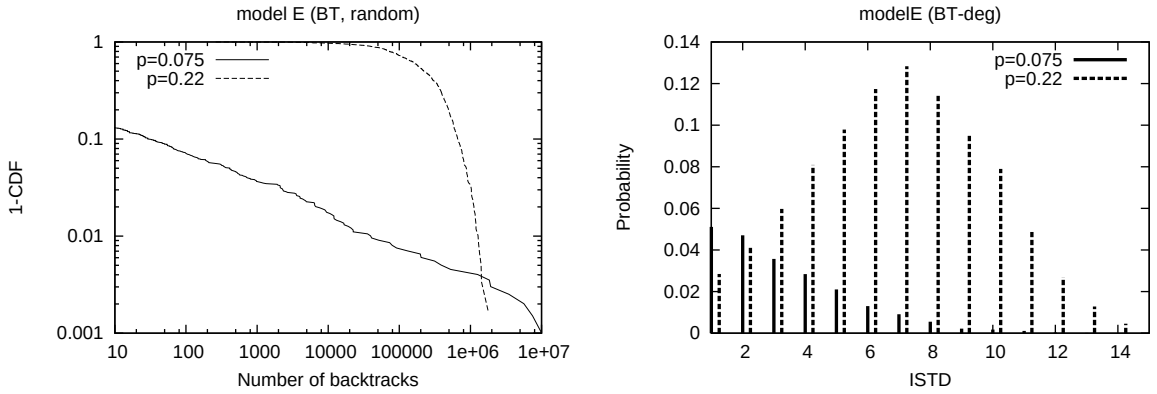

Figure 3: Example of an instance with Pareto-like ($p = 0.075$) and non-Pareto-like run time distribution ($p = 0.22$). The two instances with different regimes in decay of survival function of the run time distributions have also quite different distributions for the corresponding Inconsistency Sub-tree Depth (ISTD). For Pareto-like distributions of the run time, the corresponding ISTD follows an exponential distribution (right panel, $p = 0.075$), which is not the case when the run time distribution has exponential decay ($p = 0.22$).

probability of finding inconsistencies with a few variable assignments, and this probability decreases exponentially as the variable assignments increase. Contrast this behavior with the case in which the survival function of run time distribution of the backtrack search method has an exponential decay (see Figure 3, left panel, $p = 0.22$). In this case, the distribution of the depth of inconsistent trees no longer decreases exponentially (see Figure 3, right panel, $p = 0.22$).

The rest of the paper is organized as follows: in next section we present definitions concerning the random CSP models and concepts used in the paper. We then present empirical results illustrating the different statistical regimes of the run time distributions of our backtrack search methods. We also provide experimental evidence of the different shapes of the distributions of the depth of inconsistent sub-trees, closely related to the tail regime of the underlying run time distributions of the backtrack search method. Finally, conclusions and future work are discussed.

2 Preliminaries

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 e 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 of a constraint network or proving that none exists.

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 $< N, D, p1, p2 >$, 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). Problems classes in this model are denoted by $< N, D, c, t >$. 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 has been proposed to overcome this weakness. It is a three parameter model, $< N, D, p >$, 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. Note, however, that there is another way of tackling the problem of flawed variables. In [14] it is shown that some properties on the relative values of N, D, p_1, and p_2, guarantee that the model is sound and scalable, for a certain range of values of the parameters.

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 subtree that does not contain any node u such that $I(u)$ is a solution. The depth of an inconsistent subtree is referred to as ISTD. We denote by $T(A, P)$ the search tree of a backtrack search algorithm A solving a particular problem P, which contains a node for each instantiation visited by A until it reached a solution or proved inconsistency of P. 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. The *search cost* of an algorithm A on a particular problem P is the number of total backtracks in $T(A, P)$.

Algorithms

In the following, we will use 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 ordering variables: 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`).

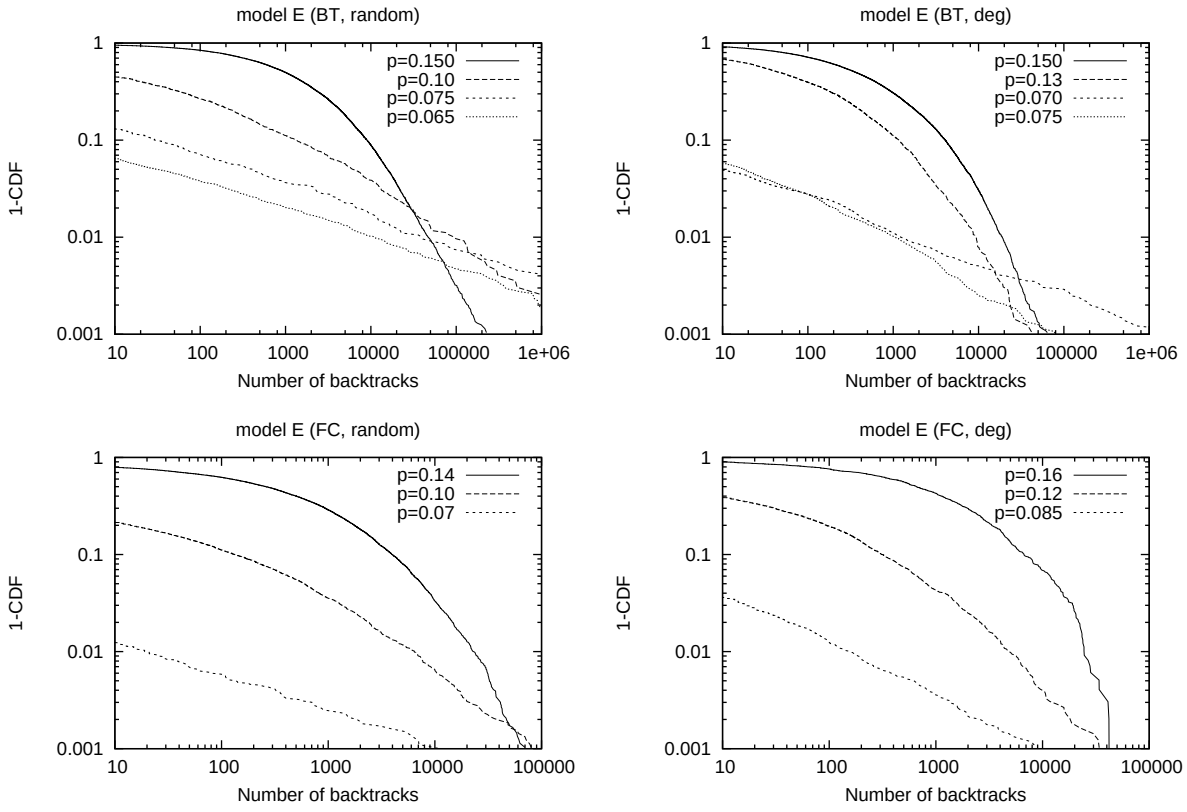

Figure 4: Survival functions of the number of backtracks for some instances generated under model E and solved using different algorithms. All the instances belong to the class $\langle 20, 10, p \rangle$.

Pareto-like Distributions

The runtime distributions of backtrack search methods are often characterized by very long tails or *heavy-tails* (HT). These non-standard probability distributions were first introduced by Vilfredo Pareto in 1897 in the context of income distribution, and have recently received much attention because of their suitability to model stochastic phenomena subject to extreme fluctuations.

Given 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. These distributions have infinite variance when $1 < \alpha < 2$ and infinite mean and variance when $0 < \alpha <= 1$. If $1 - F(x)$ (the survival function) is plotted in a wide-ranged plot, a perfect L shape is observed. The log-log plot of the survival function of a Pareto-like distribution shows linear behavior with slope equal to $-\alpha$.

3 Empirical Results

In the previous section we formally defined our models and algorithms, as well as the concepts that are key 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. In this section we show how the behavior of these two distributions is highly correlated.

The results presented in this paper concern mainly instances of Model E, using BT and FC algorithms with different heuristics. Nevertheless, we also show some results for harder problems of model B, when using more sophisticated propagation (`MAC`). We present results for the survival functions of the search cost (number of backtracks) of our backtrack search algorithms. All the plots were computed with over at least 2000 independent executions of a given problem instance. We also present the results for the corresponding inconsistency sub-tree depth distributions (ISTD). The ISTD distributions were computed according to the following procedure:

1. At every node of the search, we translate our CSP problem in conjunction with the already assigned variables, into a SAT problem, in order to use a fast complete SAT-solver. Then we apply `satz` [9, 8] to determine if the remaining problem is consistent.[1]

2. If so, we proceed as in step 1. If not, we have found an inconsistent sub-tree (IST). In order to compute its depth, we mark the current node and proceed with the backtrack search procedure that is the focus of the stufy (e.g., pure backtrack search) until it reaches inconsistency.

3. At this point one inconsistent sub-tree (IST) has been computed. Then, we backtrack up to the marked node in step 2 and proceed as in step 1.

This procedure allows us to compute the ISTD independently of the CSP algorithm employed and skipping all the search inside an IST.

Figure 4 plots the survival functions of the number of backtracks for different instances generated under model E and solved using different algorithms (`BT-random, BT-deg, FC-random` and `FC-deg`). All the instances for model E belong to the class $\langle 20, 10, p \rangle$.

The results in Figure 4 show that the threshold for heavy-tailed behavior (*i.e.*, Pareto-like behavior with power law decay) occurs at different levels of constrainedness, depending on how powerful the propagation and the heuristic are. For example, using a simple `BT-random` algorithm, problems with $p < 0.075$ show Pareto-like distributions, whereas the larger is p beyond this point, the clearer is the exponential drop of the distribution. Considering now `FC-deg`, the threshold for heavy-tailed (Pareto-like) behavior moves closer to the phase-transition, around $p = 0.085$.

[1] Satz outperforms other available CSP solvers on these instances. One could use any fast CSP solver.

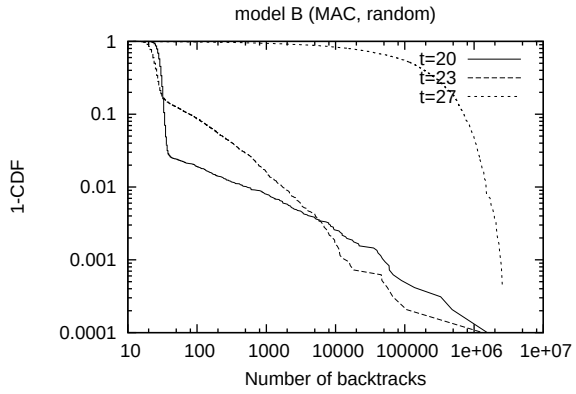

Figure 5: Survival functions of the number of backtracks for some instances generated under model B and solved with `MAC random` algorithm for problem class $\langle 50, 8, 150, t\rangle$.

Table 1: Model E $\langle 20, 10, p\rangle$. Estimation of α and its corresponding mean square error.

Algorithm	p	α	Error
`BT-random`	0.065	0.302	0.015
`BT-random`	0.075	0.318	0.009
`BT-deg`	0.070	0.418	0.004
`BT-deg`	0.075	0.448	0.026
`FC-random`	0.070	0.365	0.003
`FC-random`	0.100	0.506	0.053
`FC-deg`	0.085	0.517	0.009
`FC-deg`	0.120	0.588	0.068
`FC-dom+deg`	0.120	0.915	0.031
`FC-dom+deg`	0.140	0.833	0.080

We also observed such a clear separation of two distinct statistical regimes – Pareto-like distribution of the run-time distributions vs. non-Pareto-like distributions – for instances of model B, for different problem sizes and with more sophisticated algorithms. Figure 5 shows the survival functions of run time distributions of instances of model B $\langle 50, 8, 150, t\rangle$, for different levels of constrainedness, solved with `MAC-random`. Again, the two different statistical regimes of the survival functions are quite clear.

In order to quantify the heavy-tailedness of our Pareto-like distributions we used a QQ-estimator [11]. We estimate the paramter α of our distributions by estimating the slope of the tails using a linear regression of the logarithmic value of the data against the logarithmic values of the probability. Table 1 summarizes the results for model E. The table shows instances of model E in the class $\langle 20, 10, p\rangle$. This table illustrates the following behaviors:

- The heavy-tailedness of the distributions decreases (α increases) as constrainedness increases (p increases);
- The heavy-tailedness of the distributions decreases (α increases) when using more sophisticated search algorithms (e.g., `BT-deg` exhibits less heavy-tailed behavior (smaller α) than `BT-random`.)

These results suggest that the existence of heavy-tailed behavior in the cost distributions depends on the efficiency of the heuristic and the pruning mechanisms of the backtrack search

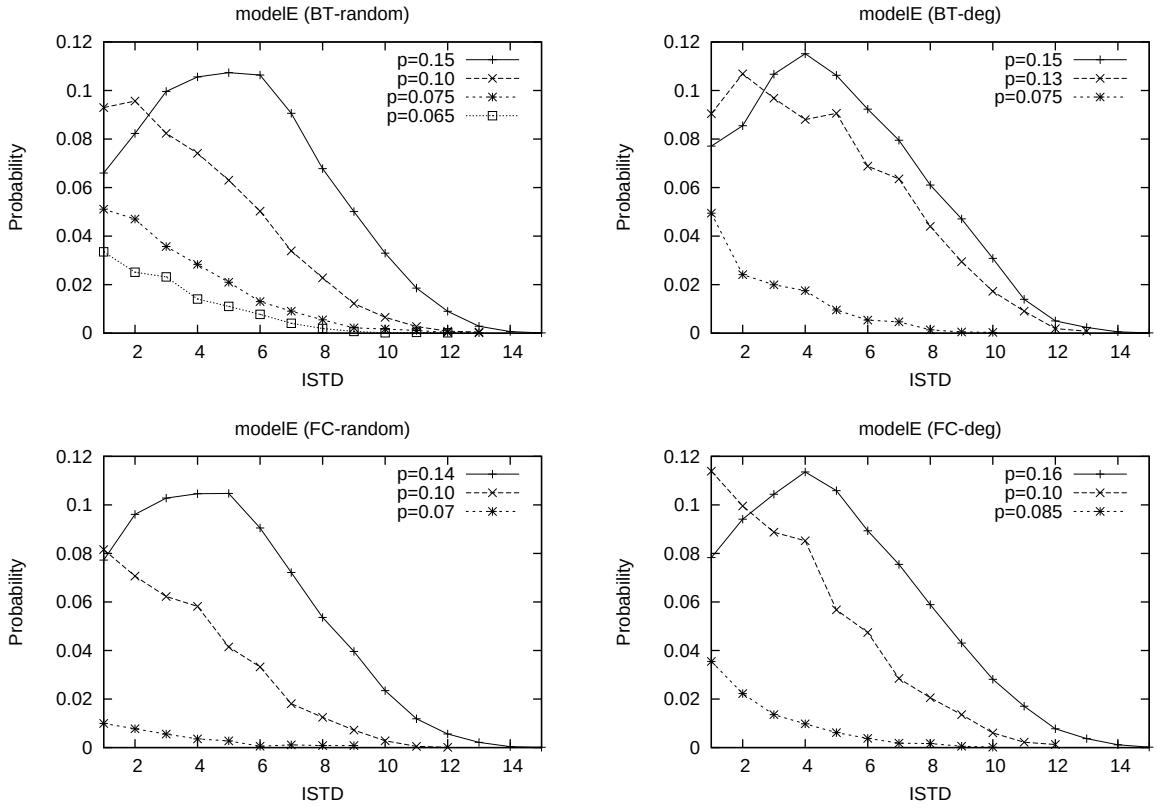

Figure 6: Model E. Distribution of the ISTD for some of the instances plotted in Figure 4.

algorithm as well as on the level of constrainedness of the problem, and of course on the size of the instances. Increasing the algorithm efficiency tends to shift the heavy-tail threshold closer to the phase transition.

A second set of results is related to the distributions of the inconsistency sub-tree detph (ISTD). Figure 6 plots the ISTD distributions for some of the instances plotted in Figure 4. It can be observed that when the cost distribution is Pareto-like, its corresponding ISTD distribution shows an exponential decay. Preliminary studies (out of the scope of this paper) explain this fact from an analytical point of view. On the other hand, as the cost distribution is less Pareto-like, its ISTD distribution moves away from an exponential decay. This effect can be observed clearly in Figure 7.

4 Conclusions and Future Work

We study the run time distributions of complete backtrack search methods on instances of well-known random CSP binary models. Our results clearly reveal different regimes in the runtime distributions of the backtrack search procedures and corresponding distributions of the depth of the inconsistent sub-trees. In the first region (this region starts at the most under-constrained area), randomized backtrack search solvers exhibit heavy-tailed behavior. The boundary of this region corresponds to a threshold for a given backtrack search algorithm. After this region, the instances become harder for the backtrack search algorithm, all the runs become homogeneously long, and therefore the variance of the backtrack search algorithm

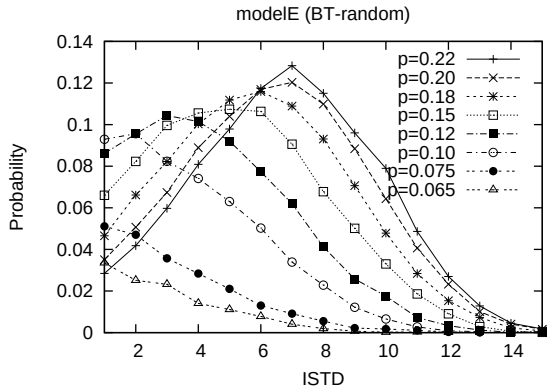

Figure 7: Distribution of the ISTD for some model E instances solved with `BT-random`.

decreases and the tails of its run time distribution decay exponentially. The location where the transition from the heavy-tailed region to the non-heavy-tailed region occurs depends on the constrainedness of the instances and the efficiency of the propagation and search heuristics of the backtrack search algorithm. The more efficient the backtrack search algorithm, the closer to the phase-transition is the heavy-tail threshold. We also show that there is a clear correlation between the regime of the tail of the run time distributions and the distributions of the depth of the inconsistent sub-trees encountered by the backtrack search method. We believe that we can exploit this correlation to design more efficient restart strategies of randomized backtrack search methods.

References

[1] D. Achlioptas, L. M. Kirousis, E. Kranakis, D. Krizanc, M. S. O. Molloy, and Y. C. Stamatiou. Random constraint satisfaction: A more accurate picture. In *Principles and Practice of Constraint Programming*, pages 107–120, 1997.

[2] H. Chen, C. Gomes, and B. Selman. Formal models of heavy-tailed behavior in combinatorial search. In *Proc. of 7th Int. Conf. of Constraint Programming CP 2001*, pages 408–422, 2001.

[3] I. Gent, E. MacIntyre, P. Prosser, B. Smith, and T. Walsh. Random constraint satisfaction: flaws and structure. In *Constraints, 6(4)*, pages 345–372, 2001.

[4] I. Gent and T. Walsh. Easy Problems are Sometimes Hard. *Artificial Intelligence*, 70:335–345, 1993.

[5] C. P. Gomes, B. Selman, and N. Crato. Heavy-tailed Distributions in Combinatorial Search. In *Proc. of the 3rd Int. Conf. of Constraint Programming (CP-97)*, pages 121–135, Linz, Austria., 1997. Springer-Verlag.

[6] C. P. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *J. of Automated Reasoning*, 24(1–2):67–100, 2000.

[7] T. Hogg, B. Huberman, and C. Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81 (1-2):1–15, 1996.

[8] Chu Min Li. A constraint-based approach to narrow search trees for satisfiability. *Information Processing Letters*, 71(2):75–80, 1999.

[9] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *IJCAI (1)*, pages 366–371, 1997.

[10] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of The Tenth National Conference on Artificial Intelligence*, pages 459–465. AAAI Press, 1992.

[11] R. Resnick. Heavy tail modelling and teletraffic data. *Annals of Statistics*, 25:1805–1869, 1997.

[12] B. Smith and S. Grant. Sparse constraint graphs and exceptionally hard problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 646–651. AAAI Press, 1995.

[13] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proceedings of the International Joint Conference on Artificial Intelligence (to appear)*, Acapulco, Mexico, 2003. AAAI Press.

[14] K. Xu and W. Li. Exact phase transition in random constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 12:93–103, 2000.