Computers & Operations Research 36 (2009) 2376 – 2386

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

A generative power-law search tree model

Alda Carvalho[a,*], Nuno Crato[b], Carla Gomes[c]

[a]Instituto Superior de Engenharia de Lisboa and Cemapre, Portugal
[b]Cemapre, Institute for Economics and Management, Technical University of Lisbon, Portugal
[c]Cornell University, Ithaca, NY, USA

ARTICLE INFO

Available online 30 August 2008

Keywords:
Random algorithms
Heavy-tails
Computational statistics

ABSTRACT

It is now a well-established fact that search algorithms can exhibit heavy-tailed behavior. However, the reasons behind this fact are not well understood. We provide a generative search tree model whose distribution of the number of nodes visited during search is formally heavy-tailed. Our model allows us to generate search trees with any degree of heavy-tailedness. We also show how the different regimes observed for the runtime distributions of backtrack search methods across different constrainedness regions of random CSP models can be captured by a mixture of the so-called stable distributions.

1. Introduction

In the last decade we have witnessed tremendous progress in the design and development of complete, backtrack-style search algorithms, for constraint satisfaction problems (CSPs). For example, in Boolean satisfiability (SAT) problems, while in the early 1990s we could only solve formulas with around 100 variables and 1000 clauses, current state-of-the-art complete DPLL-based SAT solvers can now handle much larger real-world instances, with over 1,000,000 variables and over 5,000,000 constraints. Several factors have contributed to such a progress, in particular more sophisticated data-structures, non-chronological backtracking, fast pruning and propagation methods, nogood (or clause) learning, and more recently randomization and restarts.

Randomization has greatly extended our ability to solve hard computational problems. In general, however, we think of randomization in the context of local search. While local search methods have proven to be very powerful, in some situations they cannot supplant complete or exact methods due to their inherent limitation: local search methods cannot prove inconsistency or optimality. Somehow surprisingly, randomization and restarts have also been shown quite effective for complete backtrack-style search methods. See e.g., [1–7]. In fact, randomization and restarts are now an integral part of most state-of-the-art complete SAT and CSP[1] solvers. See e.g., [9–14].

The discovery of the effectiveness of randomization and restart strategies in complete or exact search methods was made in the context of the study of the runtime distributions of backtrack-style algorithms. For a long time researchers had observed that the performance of backtrack-style search methods can vary dramatically depending on the way one selects the next variable to branch on (the "variable selection heuristic") and on what order the possible values are assigned to a variable (the "value selection heuristic"). In fact, quite often the branching heuristics provide incorrect search guidance, forcing the procedure to explore large subtrees of the search space that do not contain any solution. As a consequence, backtrack-search methods exhibit a large variance in performance. For example, we see significant differences on runs of different heuristics, runs on different problem instances, and, for randomized backtrack-search methods, runs with different random seeds. The inherent exponential nature of the search process appears to magnify the unpredictability of search procedures. In fact, it is not uncommon to observe a backtrack-search procedure "hang" on a given instance, whereas a different heuristic, or even just another randomized run, solves the instance quickly.

Even though researchers had been well aware of the high variance of backtrack-search algorithms, the discovery of the so-called heavy-tailed nature of the runtime distributions of backtrack-search methods was somehow surprising and even counter-intuitive [1,15–17]. Heavy-tailed distributions exhibit power-law decay of the tails. That is why heavy-tailed distributions are also referred to as *power-laws*. The power-law decay of the tail causes it to be *heavy* and therefore some of the moments do not converge—heavy-tailed distributions are therefore characterized by infinite moments, e.g., they can have infinite mean, or infinite variance, etc. This is in contrast with

* Corresponding author.
E-mail addresses: acarvalho@dem.isel.ipl.pt (A. Carvalho), ncrato@iseg.utl.pt (N. Crato), gomes@cs.cornell.ed (C. Gomes).

[1] The new version of ILOG Solver will include randomization and restarts [8].

0305-0548/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2008.08.017

non-heavy-tailed distributions characterized by exponential decay. Related to *heavy*-tailedness is *fat-tailedness*. The notion of *fat-tailedness* may be introduced using the concept of *kurtosis*,[2] and comparing the *kurtosis* of a given distribution with the kurtosis of the standard normal distribution. The *kurtosis* of the standard normal distribution is 3. A distribution with a *kurtosis* larger than 3 is *fat-tailed* or *leptokurtic*. Like *heavy*-tailed distributions, *fat*-tailed distributions have long tails, with a considerable mass of probability concentrated in the tails. Nevertheless, the tails of *fat*-tailed distributions are lighter than *heavy*-tailed distributions. Therefore, contrarily to heavy-tailed distributions, all the moments of *fat*-tailed distributions are finite. Examples of distributions that are characterized by *fat-tails* are the exponential distribution and the lognormal distribution. Interestingly, in the context of search, heavy-tails have been observed not only in aggregated runtime distributions of backtrack-search methods, when considering a collection of instances of the same class (e.g., random binary CSP instances generated with the same parameter space), but *also* when running a randomized backtrack-search procedure on the *same* instance several times, in which the *randomization is only used to break ties in the variable and/or value selection* [1,3,15,18–20].

The understanding of the fat-tailed and heavy-tailed nature of the distributions underlying backtrack-search methods has led to the design of new search strategies, in particular restart strategies for complete backtrack-search methods. For example, Gomes et al. showed how randomized *restarts* of search procedures can dramatically reduce the variance in the search behavior. In fact, they demonstrated that a search strategy with restarts provably eliminates *heavy* tails [3]. Interestingly, Beame et al. [5] showed that clause learning combined with restarts, as used by current-state-of-the-art SAT solvers, corresponds to a proof system exponentially more powerful than that of DPLL [21,22].

While heavy-tailed behavior has been observed in backtrack-search methods, it is clear that it does not occur in all problem instances. In fact, backtrack-style algorithms exhibit dramatically different statistical regimes across the different constrainedness regions of random CSP models—a heavy-tailed regime in the under-constrained area is replaced by a non-heavy-tail regime as one moves towards the phase transition [19,23].

A deep understanding of heavy-tailed phenomena involves formal generative models. In fact, the search for good generative models for power-law distributions is a new active research area across different domains (see e.g., [24]). For example, the so-called model of preferential attachment that generates power-law degree distributions for random graphs is an abstraction for modeling how social networks or the Internet lead to heavy-tailed behavior.

The generation of power-law distributions for backtrack search is also quite challenging, especially if one attempts to capture the full behavior of backtrack search. A compromise is to produce more abstract models, such as the model proposed by Chen et al. [25]. In such a model only high level branching decisions leading to "subtrees of the search space" are modeled. Branching decisions within a given "subtree" are not modeled. Despite its level of abstraction, the model provided interesting insights into search algorithms. For example, it led to the so-called notion of backdoor set, a set of critical variables that captures the combinatorics of the problem with respect to the propagation procedure of the solver: once values are assigned to the backdoor set, the remaining problem is solved by propagation [26,27].

Our contribution in this paper is twofold. We will show how the different regimes observed in backtrack-search methods across different constrainedness regions of random CSP models can be captured by a mixture of the so-called stable distributions. Stable distributions capture a range of heavy-tailed and non-heavy-tailed distributions. We then provide a generative search tree model whose distribution of the number of nodes visited during search is formally heavy-tailed. Even though our model is an abstraction of backtrack search, it is more realistic than previous models. In particular, while the model by Chen et al. [25] only considers high level branching decisions leading to "subtrees of the search space", more specifically, subtrees of size $2^0, 2^1, 2^2, \ldots, 2^n$ nodes, our model considers finer grained branching decisions, at every node. Furthermore, it allows us to generate search trees with any degree of heavy-tailedness. As we will see, our model also captures a key aspect of heavy-tailed behavior in backtrack search—the longer the run the more unlikely it is for the search procedure to stop. This overall behavior is achieved by the fact that the probability of going down the search tree decreases exponentially, combined with the fact that, as one goes down the search tree, the probability of making a "wrong decision"—i.e., not picking a terminal node that corresponds to a solution or that leads to a proof of unsatisfiability—given all the "wrong decisions" so far, increases. These two opposite factors—an overall exponential decrease in going down the search tree and an exponential increase in search space as we go down the search tree—are key to the generation of power-law decay.

The structure of the paper is as follows. In Section 2 we define heavy-tailed and stable distributions. In Section 3 we show how a mixture of stable distributions captures the different tail regimes of backtrack-search methods across different constrainedness regions of random CSP models. In Section 4 we present a generative search tree model and formally show that the distribution of the number of nodes visited during search is heavy-tailed. In Section 5 we discuss our results and their implications.

2. Heavy-tailed and stable distributions

Many standard probability distributions, such as the normal distribution, have exponentially decreasing tails,

$$P(X>x) \sim Ce^{-x^2}, \quad C>0 \tag{1}$$

which means that outliers are very rare.

Heavy-tailed distributions were first introduced by Pareto in the context of income distributions and were extensively studied by Lévy. Until Mandelbrot's work on fractals these types of distributions, also called *power-law* distributions, were often considered pathological cases. Recently, heavy-tailed distributions have been used to model phenomena in areas as diverse as economics, physics, geophysics, biology and more recently in computer science (for a review of the literature see e.g., [24,35]).

Heavy-tailed distributions have tails that are asymptotically[3] of the Pareto–Lévy form

$$P(X>x) \sim Cx^{-\alpha}, \quad x>0 \tag{2}$$

Informally, the tails of a heavy-tailed distribution fall according to the power α, which is much slower than exponential decay. One or both tails of these distributions have power-law or hyperbolic decay. Without loss of generality, we will discuss the right tail behavior and assume that the distribution has support on the positive half line only. The log–log plot of the tail of the survival function[4] of a

[2] The *kurtosis* is defined as μ_4/μ_2^2 (μ_4 corresponds to the fourth central moment about the mean and μ_2 is the second central moment about the mean, i.e., the variance). Notice that the *kurtosis* is independent of the location and scale parameters of a distribution. If a distribution has a high central peak and long tails, then the kurtosis is in general large.

[3] $f(x) \sim g(x)$ denotes that $\lim_{x\to\infty} f(x)/g(x) \to 1$.

[4] The survival function corresponds to $P[X>x]$, also called the *complementary cumulative distribution* function, i.e., $1-F(x)$ or 1-CDF.

heavy-tailed distribution exhibits linear behavior with slope determined by α.

A random variable X is said to be *stable* if the distribution of the sum of independent copies of it has the same shape as the distribution of X. Formally, a random variable X is said to have a stable distribution if for any $n>1$, there is a positive C_n and a real number D_n such that

$$X_1 + X_2 + \cdots + X_n \stackrel{D}{=} C_n X + D_n \tag{3}$$

where $X_1, X_2, \ldots, X_n$ are independent copies of X.

Following the notation of Samorodnitsky and Taqqu [30], a stable in $\mathbb{R}$ can be represented as a distribution with four parameters

$$S_\alpha(\sigma, \beta, \mu) \tag{4}$$

where $0<\alpha\leqslant 2$is the index of stability or characteristic exponent, $\sigma>0$the scale parameter, $-1\leqslant\beta\leqslant 1$the skewness parameter and $-\infty<\mu<+\infty$the shift parameter.

Another definition is based on the characteristic function. A random variable X is said to have a stable distribution if there are parameters $0<\alpha\leqslant 2$, $\sigma\geqslant 0$, $-1\leqslant\beta\leqslant 1$ and $\mu\in\mathbb{R}$ such that its characteristic function has the following form:

$$E[\exp\{i\theta X\}] = \begin{cases} \exp\left\{-\sigma^\alpha|\theta|^\alpha \left(1 - i\beta(\operatorname{sign}\theta)\tan\frac{\pi\alpha}{2}\right) + i\mu\theta\right\} & \text{if } \alpha\neq 1 \\ \exp\left\{-\sigma|\theta|\left(1 - i\beta\frac{2}{\pi}(\operatorname{sign}\theta)\ln|\theta|\right) + i\mu\theta\right\} & \text{if } \alpha = 1 \end{cases} \tag{5}$$

and

$$\operatorname{sign}\theta = \begin{cases} 1 & \text{if } \theta>0 \\ 0 & \text{if } \theta=0 \\ -1 & \text{if } \theta<0 \end{cases}$$

The probability densities of α-stable random variables are not known in closed form, except for the following cases: Gaussian or normal distribution, $S_2(\sigma,0,\mu)=N(\mu,\sigma^2)$; Cauchy distribution, $S_1(\sigma,0,\mu)$; Lévy distribution, $S_{1/2}(\sigma,1,\mu)$; and a constant μ which has the degenerate distribution $S_\alpha(0,0,\mu)$. For a complete treatment of stable distributions see Samorodnitsky and Taqqu [30].

For stable distributions the existence of moments depends on the parameter α. When α=2 all the moments are finite. That is the case of the normal or Gaussian distribution. For $\alpha<2$, moments of X of order less than α are finite while all higher order moments are infinite, i.e., $\alpha=\sup\{b>0 : E|X|^b<\infty\}$. So, when $1<\alpha<2$, the distribution has finite mean but infinite variance. With $\alpha\leqslant 1$, the distribution has infinite mean and variance. The α in Eq. (2) is called the index of stability of the distribution. The lower the index, the heavier the tail.

3. Mixtures of distributions in random binary CSP

Random problem distributions, such as random K-SAT or random binary CSP, have played a key role in the study of typical case complexity of combinatorial search problems. Using such random problem distributions researchers have been able to provide a finer characterization of computational hardness, beyond the worst-case exponential complexity notion implied by NP-Completeness [28].

A binary CSP is defined as a set of n *variables* $\mathscr{X}=\{x_1,\ldots,x_n\}$, a set of *domains* $\mathscr{D}=\{D(x_1),\ldots,D(x_n)\}$, where $D(x_i)$ is the finite set of possible *values* for variable x_i, and a set $\mathscr{C}$ of binary *constraints* between pairs of variables. A constraint C_{ij} on an 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* to a constraint set is an instantiation of the variables such that all the constraints are satisfied. The CSP involves finding a solution to all constraints or proving that none exists. Several models for generating random CSP distributions have been proposed over the years.

Fig. 1 illustrates the typical easy–hard–easy pattern of computational hardness for random CSP instances, generated using the so-called model E. Model E is a three parameter model, $\langle N,D,p\rangle$, where N is the number of variables, D the size of the domains, 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. In the figure we consider random instances of model E with $N=17$ and $D=8$ for different values of p. The figure shows the mean computational solution cost (normalized) and the percentage of satisfiable instances, as a function of p, the constrainedness of the model. Clearly, there are different constrainedness regions: (1) the under-constrained region, for low values of p, in which most of the instances are satisfiable; in this region it is relatively easy for the search procedure to find a solution; (2) the over-constrained area, corresponding to high values of p, in which most of the instances are not satisfiable; in this region it is also relatively easy for the search procedure to prove that there is no feasible solution; and (3) the critically constrained area, the region in between the under-constrained and over-constrained area, in which computational cost peaks. The peak of the computational cost is aligned with the *phase transition* between a phase in which most of the instances are satisfiable and a phase in which most of the instances are unsatisfiable. At the phase transition, $p=0.25$, 50% of the instances are satisfiable (unsatisfiable).

Interestingly, runtime distributions also exhibit different statistical regimes along the different constrainedness regions [23]. Fig. 2 provides a detailed view of the different regimes—from heavy-tailed to non-heavy-tailed—for Model E. The figure displays the survival function, i.e., $P(X>x)$, in 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 different values of p, the constrainedness of the instances, $0.05\leqslant p\leqslant 0.24$. We clearly identify a heavy-tailed region in which the log–log plot of the survival functions exhibit linear behavior, while in the non-heavy-tailed region the drop of the survival functions is much faster than linear.

In order to model the different regimes observed along the different constrainedness regions—from a region where the runtime distributions exhibit power-law decay to a region in which the tails are fat but no longer heavy—we considered two different approaches: (1) sums of Poisson arrivals and (2) mixtures of α-stable distributions.

We remark that mixtures of exponential distributions have been shown to be well suited to model the runtime distributions of local search methods [29]. Nevertheless, in the case of complete backtrack search methods, such mixtures do not capture well the region in which the runtime distributions are heavy-tailed since exponential distributions have a tail decay much faster than power-law.

3.1. Sums of Poisson arrivals

In our first approach to model the different statistical regimes observed in backtrack search across different constrainedness regions of random binary CSP instances we considered sums of random variables involving arrival times of a Poisson process, one of the standard ways of generating heavy-tailed distributions [30].

Let $N(t)$ represent the number of arrivals in the time interval $[0,t]$. The process $\{N(t), t\geqslant 0\}$ is a Poisson process with rate λ, if the inter-arrival times $\tau_{i+1}-\tau_i$, $i\geqslant 1$, are independent and exponentially distributed with mean $1/\lambda$. Let Γ_i be the ith arrival time of the Poisson process, i.e., $\Gamma_i=\sum_{j=1}^{i} e_j$, where the e_j's are independent and exponentially distributed with mean $1/\lambda$. The random variables Γ_i are, therefore, dependent; they have a gamma distribution with parameter i and mean $E[\Gamma_i]=i/\lambda$ (gamma-Erlang distribution).

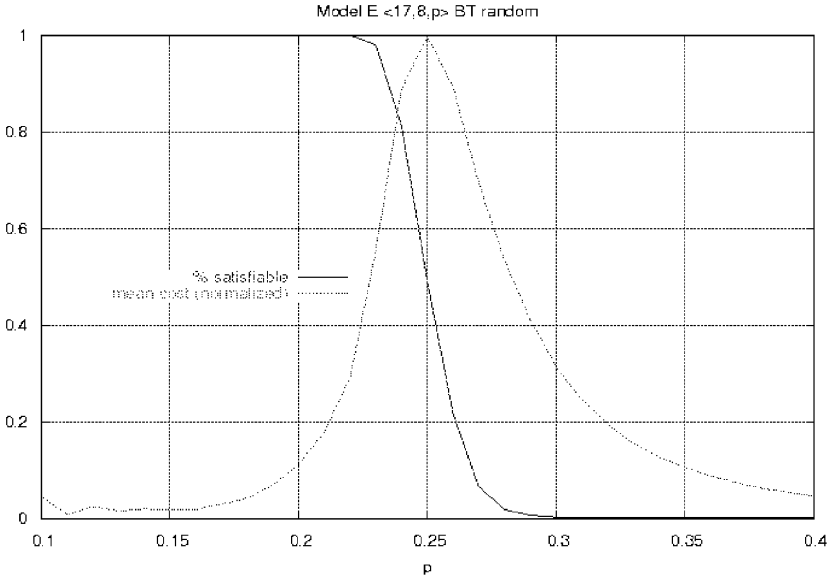

Fig. 1. Computational complexity and phase transition curves. Model E $\langle 17, 8, p\rangle$.

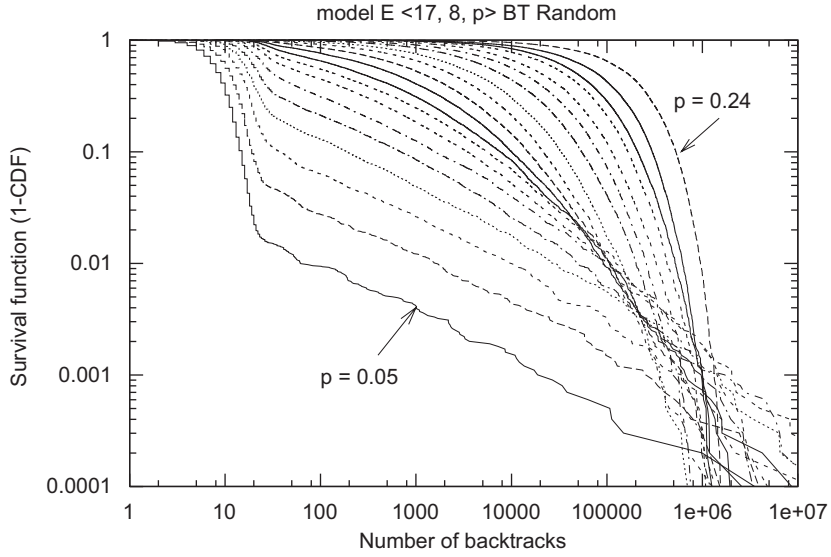

Fig. 2. The progression from heavy-tailed regime to non-heavy-tailed regime. Log–log plot of the survival functions of the runtime distributions of a backtrack-search algorithm on instances of model E $\langle 17, 8, p\rangle$, for different values of p [23].

Let $\{\tau_i\}$ denote the arrival times of a Poisson process with rate 1 and let R_i be independent and identically distributed random variables, independent of the sequence $\{\tau_i\}$. If the series

$$\sum_{i=1}^{\infty} \tau_i^{-1/\alpha} R_i \tag{6}$$

converges a.s., then it converges to a strictly α-stable random variable [30]. This result shows that an α-stable with $0 < \alpha < 2$ can be represented as a convergent sum of random variables involving arrival times of a Poisson process.

For our simulations, in Eq. (6), we use a sum to n, with τ_i i.i.d. Gamma$(i, 1)$, i integer and R_i i.i.d. $N(0, 1)$. Using $N = 10,000$ replications in the simulations, combining the number of random variables to be added (n) and tail index (α) we obtain a result as in Fig. 3. From Fig. 3 we see that when n is low, the log–log plot of the survival function exhibits linear behavior, an indication of heavy-tailed behavior. Increasing the number of random variables i $(n \uparrow)$ causes the median of the distribution to increase and the heavy-tailed regime is replaced by a fat-tailed regime, similar to the behavior observed in Fig. 2.

Despite the fact that there is a good theoretical justification for this approach, we were not able to obtain a good fit to our empirical data. The understanding of the semantics of this model involving sums of Poisson arrivals and its application to capture the different statistical regimes of backtrack-search algorithms across different constrainedness regions of random CSP instances requires further research.

3.2. Mixtures of α-stable distributions

Our most successful approach for modeling the different statistical regimes observed in backtrack search across different

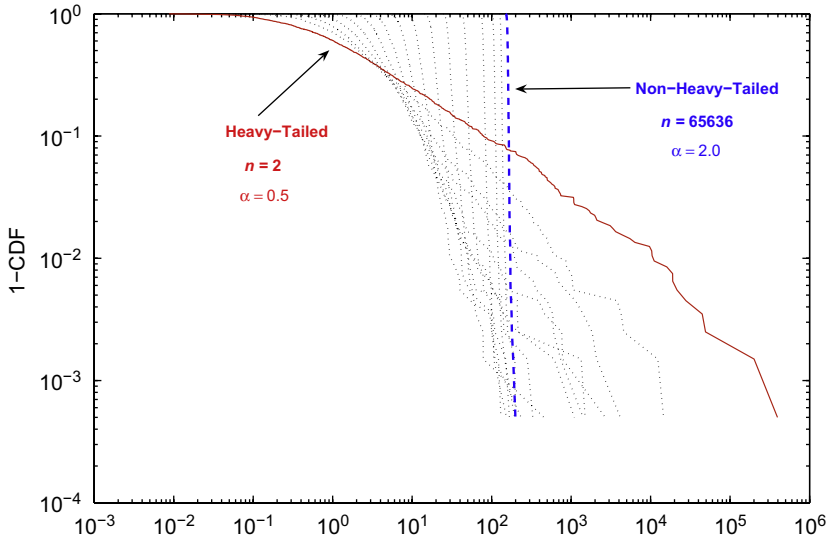

Fig. 3. Log–log plot of survival function ($n = 2^k$, $k = 1, 2, \ldots, 16$ and $\alpha = 0.5, 0.6, \ldots, 2.0$).

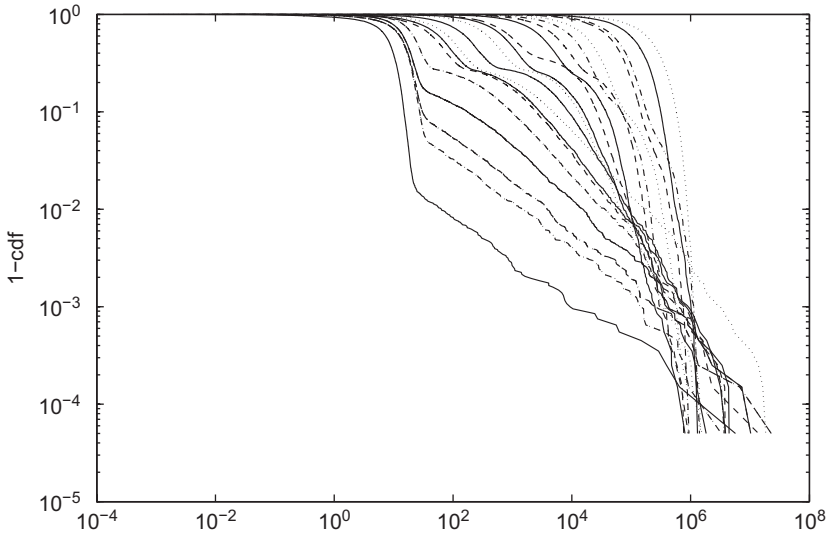

Fig. 4. Log–log plot of survival functions of stable mixtures.

constrainedness regions of random binary CSP instances was a mixture of α-stable distributions as in

$$X = \lambda S_2(\sigma_N, 0, \mu_N) + (1 - \lambda) S_\alpha(1, 0, \mu_S) \tag{7}$$

where λ is the mixture weight, $0 < \lambda < 1$.

In the mixture S_2 represents the normal component and S_α a heavy-tailed stable distribution with $\alpha < 2$. In our simulations we used N=10,000 replications. Varying simultaneously λ and the stable parameters, α, μ_N, σ_N and μ_S in (7), we were able to obtain a good fit for the different statistical regimes, as the pattern in Fig. 2. In Fig. 4 we plot the survival functions of the mixtures of stables. Interestingly, the weight of the normal component in the mixture is quite high across the different regions ($\lambda \geqslant 0.70$, except for $p = 0.16$).

The parameter α allows us to get the desired slope for the tail and to counterbalance the high weight of the normal distribution in the mixture: heavy-tails clearly appear when α is extremely low; increasing α corresponds to progressing from the heavy-tailed to the non-heavy-tailed regime. The stable parameters, μ_N and μ_S, allow us to shift the distribution. The σ_N parameter allows us to change the overall dispersion of data, which is translated on the log–log plot as the degree of smoothness on the tail transition.

The insights provided by the mixture of stables are quite interesting. The heavy-tailed regime is characterized by low values of μ_N, σ_N, μ_S and α. In this regime, the runtime distributions decay slowly—in the log–log plot we see a linear behavior over several orders of magnitude. This is due to the extremely low values of α on the heavy-tailed stable component. From an algorithmic point of view, this region corresponds to instances that are general relatively easy for the algorithm. However, now and then, the algorithm makes a sequence of wrong decisions, leading to extremely long runs. This is caused by the low values of α on the heavy-tailed stable in the mixture. When we increase the constrainedness of our model—higher p, which in our mixture model corresponds to increasing σ_N, μ_S and α—we encounter a different statistical regime in the runtime

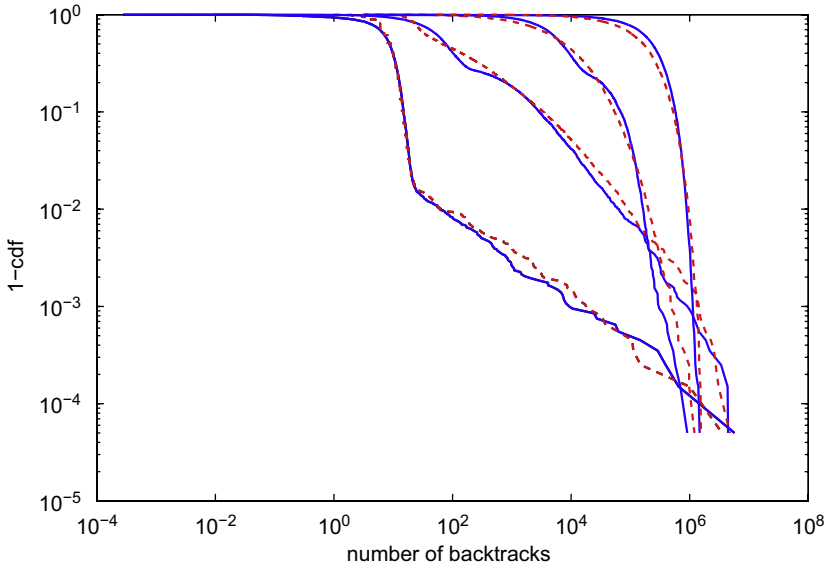

Fig. 5. Log–log plot of survival functions of stable mixtures (solid line) and empirical runtime distributions of a backtrack-search algorithm on instances of Model E, $p = 0.05$, 0.11, and 0.24 (dashed line).

distributions, where the heavy-tails disappear. In this region, the normal distribution dominates the mixture (high values of λ and α for the heavy-tailed stable component), causing the tails to drop fast. Again, from an algorithmic point of view this is quite insightful: in this region, the instances become inherently hard for the backtrack-search algorithm, all the runs become homogeneously long, the algorithm has no chance of producing short runs, therefore the variance of the backtrack search algorithm decreases and the tails of the survival functions decay rapidly.

In order to provide further evidence of the suitability of our model of mixtures of distributions, we evaluated the "goodness of fit" for all the instances of Model E discussed above. We generated the distributions using the well-tested International Mathematics and Statistics Library [31,32]. We then use an iterative procedure as described, e.g., in [33], to compare the empirical data against the fitted theoretical mixed distribution using the Kolmogorov–Smirnov (K–S) goodness of fit test. Note that we have to use such an iterative procedure given that we are using mixtures of distributions and that there is no closed form for the stable distributions with the parameters required in our model (see Section 2). The two-sample K–S test is one of the most useful and general non-parametric methods for these purposes [34]. As discussed before, given that we are interested in studying the tail behavior of the distributions, we measure the quality of the fit only taking into account tail data. For the sake of uniformity, we used 5% of the tail (500 observations) in all cases. Table 2 shows the results for the K–S statistics obtained for different instances of Model E. We observe high "p" values, which means the null hypothesis of equality of distributions is not rejected. We can thus assume our real data series are well modeled by the fitted mixture of distributions. In Fig. 5 we plot the empirical data and the fitted distribution for $p = 0.05, 0.11, 0.18$ and 0.24. The near overlap of the empirical curves and corresponding fitted distributions is striking. Our model is able to reproduce the inflexions of the plots with high accuracy.

As a final remark we would like to point out that the same qualitative behavior is observed when considering different variants of backtrack-search methods. The more sophisticated the method (e.g., nogood learning, strong pruning and propagation), the narrower the heavy-tailed region but still, as the instances become too hard, the heavy-tailed regime is replaced by a non-heavy-tailed regime.

4. Generative power-law search tree model

In this section we present an abstract generative search tree model that produces power-laws. In order to motivate the topic we start by considering two typical examples of exponential decay.

4.1. Exponential decay

We start by considering a full binary tree with N levels below the root node (see Fig. 6, left panel), therefore a total of $2^{N+1} - 1$ nodes. Let us assume we only have one terminal node (i.e., a solution node or a node that leads to a proof of unsatisfiability) and that it can be any node of the tree. We consider a search algorithm that at each branching decision picks a node with probability $\frac{1}{2}$, until the terminal node is found. We denote by X the total number of visited nodes in the tree. In this model, the probability that the search stops after n choices is 2^{-n} and

$$P(X > n) = 2^{-n} \to 0 \quad \text{as } n \to \infty,$$

which means that the tail of the distribution has exponential decay.

Now, with the same algorithm, let us consider a different binary search tree. In each level there are two equiprobable nodes, one of which is a terminal node, i.e., one in which the search terminates (a solution is found or the algorithm proves that there is no solution). (See Fig. 6, right panel.) In this case, the probability that the search stops after n choices is also 2^{-n} and the tail also has exponential decay, that is, $P(X > n) \to 0$ very quickly.

In order for us to have a model that exhibits hyperbolic or power-law tail decay, the probability of terminal nodes, as we go down the tree, must decrease, but slowly. In the following sections we present different generative models of power-law decay for the number of nodes visited during search.

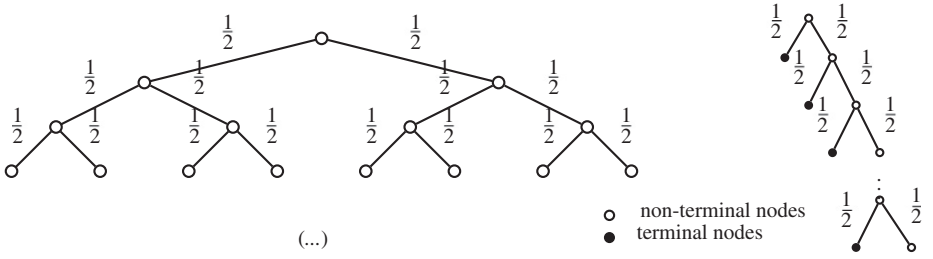

Fig. 6. Complete binary tree (left) and incomplete binary tree (right).

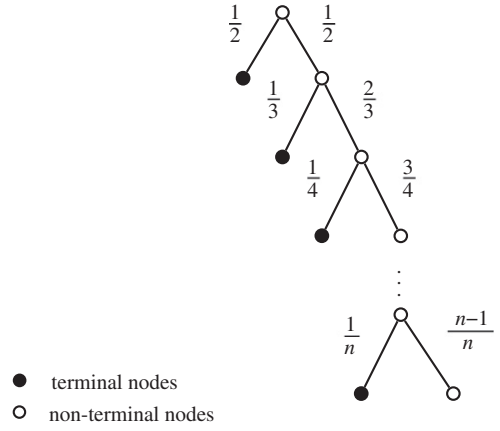

Fig. 7. Search tree with power-law decay with $\alpha = 1$.

4.2. Power-law decay

In our search tree model terminal nodes, i.e., nodes in which the search terminates (a solution is found or the algorithm proves that there is no solution) are distributed in the tree according to some rule. The search algorithm consists of descending the tree according to a random path, until a terminal node is found. The random variable, X, is the total number of visited nodes during search.

We start by considering a high level incomplete binary tree model. In this model there are two types of nodes: terminal nodes and non-terminal nodes. Terminal nodes have no children. Non-terminal nodes have two child nodes: a terminal node and a non-terminal node. Nodes are naturally organized in levels, starting with the root node at level 1. The children of the root node have level 2 and so on. As we will see, we can tune the heavy-tailedness of X by varying the probabilities of reaching a terminal node (non-terminal node) from a non-terminal node.

4.2.1. Case $\alpha = 1$

We assign probabilities to each node according to its level, l: from a non-terminal node, the probability of reaching a terminal node is $1/l$ and a non-terminal node $(1 - 1/l)$. In Fig. 7 we show the binary tree model with the corresponding branching probabilities. Again, we define X as the total number of nodes visited by a random path down the tree, until reaching a terminal node (including the terminal node). The probability mass of X is given by the following expression:

$$P(X=n) = \left(\prod_{i=2}^{n-1} \frac{i-1}{i}\right) \times \frac{1}{n} = \frac{1}{n-1} \times \frac{1}{n}, \quad n > 1$$

Proposition 1. *If the mass probability of X is*

$$P(X=n) = \frac{1}{n-1} \times \frac{1}{n} \tag{8}$$

then, for $n > 1$, $P(X > n) = 1/n$, so X has a hyperbolic or power-law decay with $\alpha = 1$.

Proof. If (8) then

$$P(X \leqslant n) = \sum_{i=2}^{n} \frac{1}{i-1} \times \frac{1}{i} = \left(\frac{1}{1} - \frac{1}{2}\right) + \left(\frac{1}{2} - \frac{1}{3}\right) + \cdots + \left(\frac{1}{n-1} - \frac{1}{n}\right) = 1 - \frac{1}{n}$$

so $P(X > n) = 1/n$.

From Eq. (2), with $\alpha = 1$ we see

$$\lim \left(\frac{Cn^{-1}}{\frac{1}{n}}\right) = C \in \mathbb{R}^+ \Longrightarrow P(X > n) \sim C\frac{1}{n} \qquad \square$$

4.3. Case $\alpha \in \mathbb{Q}$

The following result allows us to build a stable tree for any rational $\alpha \in (0, 2)$. We shall consider the same search algorithm as before. Again, we assume a binary tree (terminal vs. non-terminal node) and we set up the probabilities according to the node levels as shown in Fig. 8. The probability mass of X is given by the following expression:

$$P(X=n) = \frac{2\alpha}{2n-1+\alpha} \prod_{i=2}^{n-1} \frac{2i-1-\alpha}{2i-1+\alpha}, \quad n > 1 \tag{9}$$

Proposition 2. *If the mass probability of X is*

$$P(X=n) = \frac{2\alpha}{2n-1+\alpha} \prod_{i=2}^{n-1} \frac{2i-1-\alpha}{2i-1+\alpha}, \quad n > 1 \tag{10}$$

Then, for $n > 1$,

$$P(X > n) = \prod_{i=2}^{n} \frac{2i-1-\alpha}{2i-1+\alpha}$$

and X as a hyperbolic decay with the index of stability α.

Proof.

$$P(X > n) = \prod_{i=2}^{n} \frac{2i-1-\alpha}{2i-1+\alpha} = \frac{\alpha-1}{\alpha+1} \prod_{i=0}^{n} \frac{i + \frac{1+\alpha}{2}}{i + \frac{1-\alpha}{2}}$$

$$= \frac{\alpha-1}{\alpha+1} \prod_{i=0}^{n} \frac{i - 1 + \frac{1-\alpha}{2}}{i - \frac{1-\alpha}{2}}$$

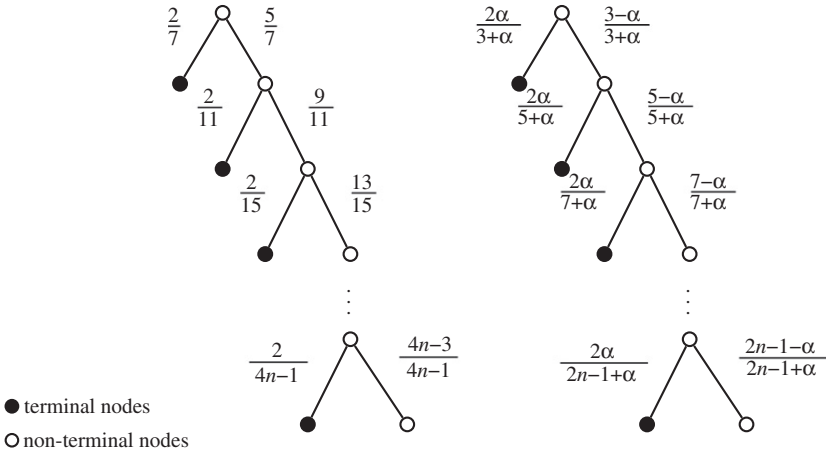

Fig. 8. Search tree with power-law decay: $\alpha = \frac{1}{2}$ (left) and $\alpha \in \mathbb{Q}$ (right).

To simplify the notation let us use $d = (1-\alpha)/2$.

$$P(X>n) = \frac{\alpha-1}{\alpha+1}\prod_{i=0}^{n}\frac{i-1+d}{i-d} = \frac{\alpha-1}{\alpha+1}\frac{\Gamma(i+d)\Gamma(i-d)}{\Gamma(i-d+1)\Gamma(d)} \tag{11}$$

Applying Stirling's formula to (11),

$$\begin{aligned}
P(X>n) &\sim \frac{\alpha-1}{\alpha+1}\frac{\Gamma(1-d)}{\Gamma(d)}\frac{\sqrt{2\pi}e^{-n-d+1}(n+d-1)^{n+d-1/2}}{\sqrt{2\pi}e^{-n+d}(n-d)^{n+d+1/2}} \\
&= e^{-2d+1}\frac{\alpha-1}{\alpha+1}\frac{\Gamma(1-d)}{\Gamma(d)}\frac{(n+d-1)^{n+d-1/2}}{(n-d)^{n-d+1/2}} \\
&= e^{-2d+1}\frac{\alpha-1}{\alpha+1}\frac{\Gamma(1-d)}{\Gamma(d)} \\
&\quad\times\frac{(n+d-1)^{n+d-1/2}}{(n-d)^{n-d+1/2}(n-d)^{-2d+1}} \\
&= e^{-2d+1}\frac{\alpha-1}{\alpha+1}\frac{\Gamma(1-d)}{\Gamma(d)}\left(1+\frac{2d-1}{n-d}\right)^{n-d} \\
&\quad\times\left(1+\frac{2d-1}{n-d}\right)^{2d-1/2}(n-d)^{2d-1} \\
&\to e^{-2d+1}\frac{\alpha-1}{\alpha+1}\frac{\Gamma(1-d)}{\Gamma(d)}e^{2d-1}(n-d)^{2d-1} \\
&= \frac{\alpha-1}{\alpha+1}\frac{\Gamma(1-d)}{\Gamma(d)}(n-d)^{2d-1} \sim Cn^{-\alpha} \qquad \square
\end{aligned}$$

4.4. Case $\alpha \in \mathbb{R}$

A generalization of the previous cases, i.e., $\forall\alpha \in (0,2)$, leads to some technicalities, namely due to the fact that the branching probabilities can be irrational numbers.

Let us consider, for example, the case $\alpha = \pi/2$, then we have $P(X>n) = 1/\sqrt{n^{\pi}}$. In this case we want to generate a tree with the behavior in Fig. 9 (left). Let us denote the sequence of probabilities for each non-terminal node level by U_n. It is not physically possible to construct the tree depicted in Fig. 9, left panel, due to the irrationality of the weights in question. As an approximation, let us construct another tree in which the sequence of the branching probabilities leading to non-terminal nodes correspond the truncated U_n sequence, with n decimal digits, denoted by $\mathrm{Trunc}_n(U_n)$. Fig. 9 (right panel) illustrates such a tree for $\alpha = \pi/2$. See Table 1 for an example of the truncation approximation.

In the ideal case, presented in Fig. 9 (left panel), we have

$$P(X>n) = \prod_{n\geqslant 2}\sqrt{\left(\frac{n-1}{n}\right)^{\pi}} = \frac{1}{\sqrt{n^{\pi}}} \tag{12}$$

and in the approximation case, presented in Fig. 9 (right panel), we have

$$P(X>n) = \prod_{n\geqslant 2}\mathrm{Trunc}_n\sqrt{\left(\frac{n-1}{n}\right)^{\pi}} \tag{13}$$

In Appendix A we show that

$$\lim\left(\frac{\prod_{n\geqslant 2}\mathrm{Trunc}_n(U_n)}{\prod_{n\geqslant 2}U_n}\right) = C \in \mathbb{R}^+ \tag{14}$$

In other words, the power-law tail decay of X, assuming branching probabilities given by the approximation $\mathrm{Trunc}_n(U_n)$, is similar to the corresponding power-law tail decay considering the original, untruncated branching probabilities, U_n.

In summary we can generate trees with any desired power-law tail decay by constructing binary trees with the branching probabilities for non-terminal nodes given by the sequence U_n or $\mathrm{Trunc}_n(U_n)$ defined as follows[5]:

Case $\alpha \in \mathbb{Q}$:

$$U_n = \frac{2n-1-\alpha}{2n-1+\alpha} \tag{15}$$

Other cases:

$$\mathrm{Trunc}_n(U_n) = \mathrm{Trunc}_n\left(\frac{n-1}{n}\right)^{\alpha} \tag{16}$$

5. Discussion

We introduced a generative model that allows us to generate search trees with any degree of heavy-tailedness. While it is only an abstract model, we believe it captures key aspects of heavy-tailed behavior in combinatorial search. Our model is also more realistic than a formal model proposed previously [25]. In that model only high level decisions are considered, leading to subtrees of size

[5] The branching probabilities for terminal nodes are the complement to one of the branching probabilities for non-terminal nodes.

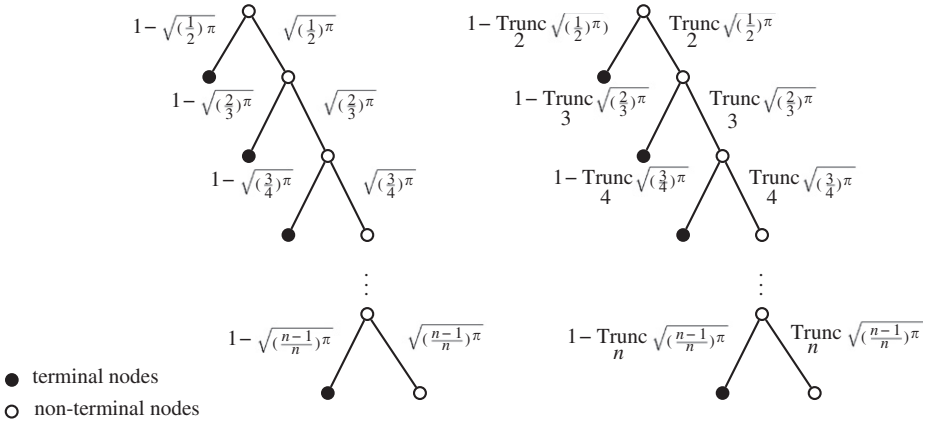

Fig. 9. Search tree with stable decay for $\alpha = \pi/2$: ideal (left) and approximation (right).

Table 1
Fit results of α-stable mixture: $X = \lambda S_2(\sigma_N, 0, \mu_N) + (1-\lambda)S_\alpha(1, 0, \mu_S)$

p	μ_N	σ_N	λ	α	μ_S	p-Value KS
0.05	9	4.5	0.9	0.5	1	0.9051
0.06	12	6	0.85	0.5	10	0.9051
0.07	15	7.5	0.8	0.5	15	0.8320
0.08	15	10	0.8	0.6	100	0.9341
0.09	20	10	0.7	0.7	1000	0.9999
0.10	25	50	0.7	0.8	1100	0.9968
0.11	30	70	0.7	0.8	1400	0.6616
0.12	35	100	0.7	0.7	2000	0.9975
0.13	50	200	0.7	1.0	5000	0.9988
0.14	0	500	0.7	1.3	8500	0.9621
0.15	0	900	0.7	1.3	10,000	0.9404
0.16	0	1000	0.5	1.7	15,000	0.8111
0.17	0	4000	0.7	1.8	27,000	0.7326
0.18	0	7000	0.7	1.9	50,000	0.8111
0.19	0	15,000	0.8	1.9	70,000	0.8970
0.20	0	27,000	0.85	1.9	120,000	0.9959
0.21	0	70,000	0.9	2.0	200,000	0.9464
0.22	0	85,000	0.9	2.0	300,000	0.3114
0.23	0	180,000	0.95	2.0	300,000	0.8004
0.24	0	350,000	0.99	2.0	300,000	0.9922

$2^0, 2^1, 2^2, \ldots, 2^n$ nodes. Our model explicitly considers branching decisions at every node of the tree, while still capturing a key aspect of heavy-tailed behavior in backtrack search—the longer the run the more unlikely it is for the search procedure to stop. This is captured by the fact that, even though the probability of going down the search tree decreases exponentially, as one goes down the search tree, the probability of not picking a terminal node that corresponds to a solution or that leads to a proof of unsatisfiability, given all the "wrong decisions" so far, increases, leading to an exponential increase in the size of the search space searched. These two opposite factors—exponential increase in search space as we go down the search tree, with an overall exponential decrease in going down the search tree—are key to the generation of power-law decay.

While we presented our model as a binary tree, it can be easily adapted to other tree shapes that resemble more the search trees produced in combinatorial search. In fact, our abstract generative binary model presented in Section 4.2 can be mapped into different tree shapes: the key issue is that from each non-terminal node the probability of reaching a terminal node in level l is $1/l$; the remaining probability $(1 - 1/l)$ is distributed among non-terminal nodes. In Fig. 10 we show two variants of the binary tree presented in Section 4.2, for the case of $\alpha = 1$. The middle case provides an example of how to construct a tree with equiprobable nodes at each level. Again,

Table 2
Truncated probabilities for a tree with equiprobable nodes at each level with $\alpha = \pi/2$

U_n	$\text{Trunc}_n(U_n)$	Level	Total of branches	Branches with non-terminal node
$\sqrt{\left(\frac{1}{2}\right)^\pi}$	0.33	2	100	33
$\sqrt{\left(\frac{2}{3}\right)^\pi}$	0.528	3	1000	528
$\sqrt{\left(\frac{3}{4}\right)^\pi}$	0.6364	4	10,000	6364
$\sqrt{\left(\frac{4}{5}\right)^\pi}$	0.70432	5	100,000	70,432
$\sqrt{\left(\frac{5}{6}\right)^\pi}$	0.750969	6	1,000,000	750,969
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\sqrt{\left(\frac{n-1}{n}\right)^\pi}$		n	10^n	$\underset{n}{\text{Trunc}}\left(\sqrt{\left(\frac{n-1}{n}\right)^\pi}\right) \times 10^n$

if we consider irrational branching probabilities, the construction of such a tree is more involved. Table 1 provides an example for the case of $\alpha = \pi/2$.

We should also point out that the nodes in our model capture different decision points such as picking the next variable to branch on, or picking a value to assign to a variable, or picking a backtracking point, or more generally picking or not picking the "right terminal node".[6] Therefore, our model can be viewed as an abstraction for different variants of backtrack-search models.

Finally, even though our results are asymptotic, we observe a clear indication of heavy-tailed behavior even for relatively small values of n. As illustrated in Fig. 11, for the case of $\alpha = 1$, while for $n = 10$, 100, and 1000 the log–log plot of tail of the survival function exhibits a sharp drop, for $n = 10,000$, $n = 100,000$, and $n = 1,000,000$ it already exhibits a close-to-linear behavior, which is a clear indication of heavy-tailed behavior. Of course, we are dealing with finite problems and these distributions have a non-finite support. The actual finite support problems would correspond to truncated distributions, which could be modeled analogously to the way done in [25].

We also showed how the different statistical regimes observed in the runtime distributions of backtrack-style algorithms on random CSP instances can be captured by a mixture of stable distributions,

[6] Note that in algorithms that perform chronological backtrack, once a dead-end is reached the algorithm automatically backtracks to the last variable considered, with remaining unexplored values, and therefore the only decision consists on picking the next value to assign to the variable.

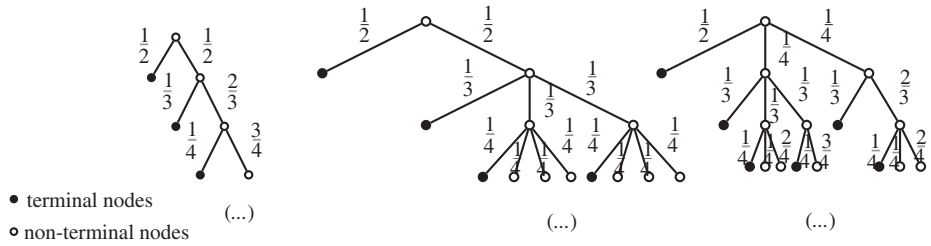

Fig. 10. Different variants of search trees with power-law decay with $\alpha = 1$.

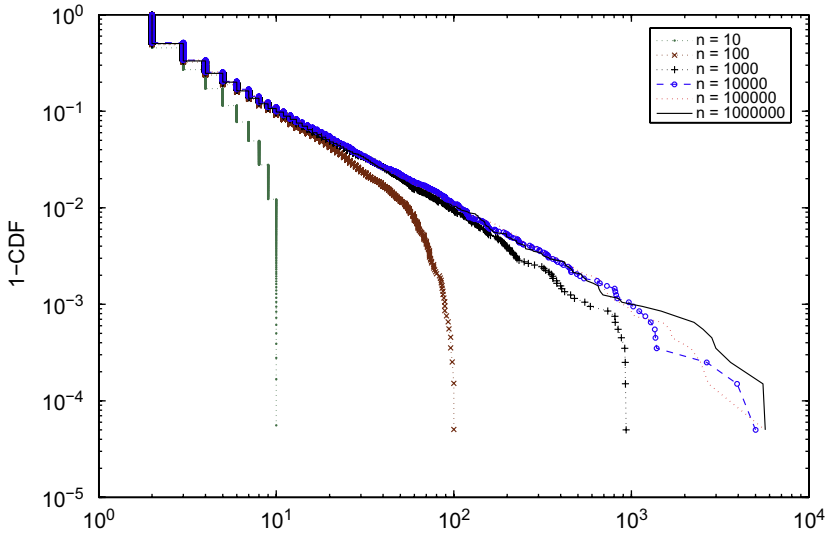

Fig. 11. Tails obtained with our generative model for $\alpha = 1$ and different values of n.

in which one of the components is heavy-tailed and the other component is the normal distribution. This mixture provides interesting insights: despite the relative high weight of the normal distribution across the different regions, the extremely low α values of the heavy-tailed stable produce a heavy-tailed regime; as α is increased, the heavy-tailed component no longer outweighs the normal component, leading to exponentially decaying tails. From an algorithmic point of view the heavy-tailed regime corresponds to having an algorithm that has good chances of finding solutions with short runs, given the fact that this regime is in the under or medium constrained area, but now and then it makes a sequence of mistakes that leads to extremely long tails, therefore heavy-tails. As the instances become harder, the heavy-tailed regime is replaced by a non-heavy-tailed regime in which the normal distribution dominates, with corresponding increase in μ. In this region the instances become inherently harder, all the runs become homogeneously long, the algorithm does not have a chance of producing short runs, therefore the dramatic decrease in the ranges of the runtime distributions and the fast drop of the tails.

In summary, we introduced a generative search tree model that captures key aspects of heavy-tailed behavior in combinatorial search. Furthermore our model allows us to generate search trees with any degree of heavy-tailedness. We also showed how a mixture of stable distributions captures the statistical regimes observed in runtime distributions of backtrack-style algorithms across different constrainedness regions of random CSP instances. We hope our models will lead to further improvements in the design of search methods.

Appendix A.

Let us denote the sequence of probabilities for each non-terminal node of level n by U_n and by $\mathrm{Trunc}_n(U_n)$ the corresponding truncated sequence with n decimal digits. Our goal is to show that using $\mathrm{Trunc}_n(U_n)$ instead of U_n leads to the same power-law decay for the distribution of X, the number of visited nodes during the search.

Lemma. *If* $U_n \to 1$, $U_n > 0$ *and* $\mathrm{Trunc}_n(U_n + 1) \geqslant \mathrm{Trunc}_n(U_n)$ *then* $\mathrm{Trunc}_n/U_n \to 1$.

Proof. Let us start to note that

$$\frac{\mathrm{Trunc}_n(U_n)}{U_n} = 1 - \frac{U_n - \mathrm{Trunc}_n(U_n)}{U_n} \leqslant 1$$

Due the nature of truncation, $U_n - \mathrm{Trunc}_n(U_n) \leqslant 10^{-n}$. Therefore,

$$1 - \frac{10^{-n}}{U_n} \leqslant 1 - \frac{U_n - \mathrm{Trunc}_n(U_n)}{U_n} \leqslant 1$$

Since

$$\lim\left(1 - \frac{10^{-n}}{U_n}\right) = 1 - \frac{\lim(10^{-n})}{\lim(U_n)} = 1$$

then

$$\frac{\mathrm{Trunc}_n(U_n)}{U_n} \to 1 \qquad \square$$

Defining the sequence

$$W_n = \prod_n U_n = \begin{cases} W_1 = U_1 \\ W_{n+1} = W_n \times U_{n+1} \end{cases}$$

and noting that

$$\ln(W_n) = -\sum \ln(U_n^{-1})$$

then

$$W_n \to 0 \;\Leftrightarrow\; \ln(W_n) = -\sum \ln(U_n^{-1}) \to +\infty \tag{17}$$

which will simplify the computations that follow.

Proposition 3. *Let us considered the sequence* $U_{n\geqslant 2} \to 1$, $U_n > 0$ *and* $\mathrm{Trunc}_n(U_n+1) \geqslant \mathrm{Trunc}_n(U_n)$, *then*

$$\lim\left(\frac{\prod_{n\geqslant 2}\mathrm{Trunc}_n(U_n)}{\prod_{n\geqslant 2}U_n}\right) = \lim \prod_{n\geqslant 2}\left(\frac{\mathrm{Trunc}_n(U_n)}{U_n}\right) = C \in \mathbb{R}^+ \tag{18}$$

Proof. By (17), to prove this theorem is analogous to show that

$$\sum \ln\left(\frac{\mathrm{Trunc}_n(U_n)}{U_n}\right)^{-1} = \sum \ln\left(\frac{U_n}{\mathrm{Trunc}_n(U_n)}\right) < +\infty \tag{19}$$

Using lemma, $U_n/\mathrm{Trunc}_n(U_n) \to 0$. Therefore, let us see if (19) converges.

Since $\ln(x) \leqslant x$, for $x > 0$, then

$$\ln\left(1 + \frac{U_n - \mathrm{Trunc}_n(U_n)}{\mathrm{Trunc}_n(U_n)}\right) \leqslant \frac{U_n - \mathrm{Trunc}_n(U_n)}{\mathrm{Trunc}_n(U_n)} \leqslant \frac{U_n - \mathrm{Trunc}_n(U_n)}{\mathrm{Trunc}_2(U_2)}$$

Since the first significant digits of $U_n - \mathrm{Trunc}_n(U_n)$ appears at the $(n+1)$ decimal place, we have $U_n - \mathrm{Trunc}_n(U_n) \leqslant 10^{-n}$. Therefore,

$$\ln\left(1 + \frac{U_n - \mathrm{Trunc}_n(U_n)}{\mathrm{Trunc}_n(U_n)}\right) \leqslant \frac{10^{-n}}{\mathrm{Trunc}_2(U_2)} \tag{20}$$

Comparing both sequences (20), since $\sum_{n=2}^{\infty} 10^{-n}/\mathrm{Trunc}_n(U_2)$ is convergent then $\sum \ln(\mathrm{Trunc}_n(U_n)/U_n)^{-1}$ is also convergent. □

A.1. In summary

Proposition 3 shows that the sequences

$$\prod_{n\geqslant 2} U_n \quad \text{and} \quad \prod_{n\geqslant 2} \mathrm{Trunc}(U_n)$$

are asymptotically of the same order. We can therefore use $\mathrm{Trunc}_n(U_n)$ to generate a search tree whose distribution X, the number of nodes visited during search, can have any desired power-law tail decay (i.e., the index of stability, α, $0 < \alpha < 2$).

References

[1] Gomes CP, Selman B, Crato N. Heavy-tailed distributions in combinatorial search. In: Proceedings of the third international conference of constraint programming (CP-97), Linz, Austria. Berlin: Springer; 1997.

[2] Gomes C, Selman B, Kautz H. Boosting combinatorial search through randomization. In: Proceedings of the 15th national conference on artificial intelligence (AAAI-98). New Providence, RI: AAAI Press; 1998. p. 431–8.

[3] Gomes CP, Selman B, Crato N, Kautz H. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. Journal of Automated Reasoning 2000;24(1–2):67–100.

[4] Lynce I, Marques-Silva J. Complete unrestricted backtracking algorithms for satisfiability. In: Proceedings fifth international symposium on theory and applications of satisfiability testing (SAT02), 2002.

[5] Beame P, Kautz HA, Sabharwal A. Towards understanding and harnessing the potential of clause learning. Journal of Artificial Intelligence Research 2004;22:319–51.

[6] Refalo P. Impact-based search strategies for constraint programming. In: Proceedings CP'04, 2004.

[7] Huang J. The Effect of restarts on the efficiency of clause learning. In: Proceedings of the 20th international joint conference on artificial intelligence (IJCAI-07), 2007.

[8] Shaw P. Personal communication, 2006.

[9] Li CM, Anbulagan A. Heuristics based on unit propagation for satisfiability problems. In: Proceedings of the international joint conference on artificial intelligence, Nagoya, Japan, 1997.

[10] Marques-Silva J, Sakallah K. Grasp—a search algorithm for propositional satisfiability. IEEE Transactions on Computers 1999;48(5):506–21.

[11] Ilog. Ilog solver, 2006.

[12] Moskewicz M, Madigan C, Zhao Y, Zhang L, Malik S. Chaff: engineering and efficient SAT solver. In: Proceedings of the 39th design automation conference, Las Vegas, 2001.

[13] Goldberg E, Novikov Y. Berkmin: a fast and robust sat-solver. In: Proceedings of design automation and test in Europe (DATE-02), 2002.

[14] Een N, Sorensson N. Minisat—a sat solver with conflict-clause minimization. In: Sat 2005 Competition, 2005.

[15] Hogg T, Williams C. The hardest constraint problems: a double phase transition. Artificial Intelligence 1994;69:359–77.

[16] Gent I, Walsh T. Easy problems are sometimes hard. Artificial Intelligence 1993;70:335–45.

[17] Smith B, Grant S. Sparse constraint graphs and exceptionally hard problems. In: Proceedings of the international joint conference on artificial intelligence. AAAI Press; 1995. p. 646–51.

[18] Walsh T. Search in a small world. In: Proceedings of the international joint conference on artificial intelligence, Stockholm, Sweden, 1999.

[19] Hulubei T, O'Sullivan B. The impact of search heuristics on heavy-tailed behaviour. Constraints 2006;11(2).

[20] Hulubei T, O'Sullivan B. Heavy-tailed runtime distributions: heuristics, models and optimal refutations. In: Proceedings CP'06, 2006.

[21] Davis M, Putnam H. A computing procedure for quantification theory. Journal of ACM 1960;7(3):201–15.

[22] Davis M, Logemann G, Loveland D. A machine program for theorem-proving. Communications of the ACM 1962;5:394–7.

[23] Gomes C, Fernandez C, Selman B, Bessiére C. Statistical regimes across constrainedness regions. Constraints 2005;10(4):317–37.

[24] Mitzenmacher M. A brief history of generative models for power law and lognormal distributions. Internet Mathematics 2004;1(2):226–51.

[25] Chen H, Gomes C, Selman B. Formal models of heavy-tailed behavior in combinatorial search. Proceedings of seventh international conference on the principles and practice of constraint programming (CP-2001). Lecture notes in computer science, vol. 2239. Berlin: Springer; 2001. p. 408–22.

[26] Williams R, Gomes C, Selman B. Backdoors to typical case complexity. In: Proceedings of the 18th international joint conference on artificial intelligence (IJCAI-03), 2003.

[27] Kilby P, Slaney J, Thiebaux S, Walsh T. Backbones and backdoors in satisfiability. In: Proceedings of the 20th national conference on AI. AAAI; 2005.

[28] Hogg T, Huberman B, Williams C. Phase transitions and complexity. Artificial Intelligence 1996;81..

[29] Hoos H. A mixture-model for the behaviour of SLS algorithms for sat. In: Proceedings of AAAI-02, 2002.

[30] Samorodnitsky G, Taqqu M. Stable non-Gaussian processes: stochastic models with infinite variance. London: Chapman & Hall; 1994.

[31] Visual Numerics Inc. International Mathematics and Statistics Library, 2007 ⟨http://www.vni.com/products/imsl/⟩.

[32] Lewis P, Orav EJ. Simulation methodology for statisticians, operations analysts, and engineers. London: Chapman & Hall, CRC; 1988.

[33] Scalas E, Kim K. The art of fitting financial time series with levy stable distributions, 2006 ⟨http://arxiv.org.abs/physics/0608224v1⟩, submitted for publication.

[34] Lindgren BW. Statistical theory. London: Chapman & Hall; 1993.

[35] Adler, R., Feldman, R., Taqqu, M.S. A practical guide to heavy tails: statistical techniques for analyzing heavy tailed distributions. Birkhäuser, Boston; 1998.