Fat- and Heavy-Tailed Behavior in Satisficing Planning

Eldan Cohen and **J. Christopher Beck**
Department of Mechanical & Industrial Engineering
University of Toronto
Toronto, Canada
{ecohen, jcb}@mie.utoronto.ca

Abstract

In this work, we study the runtime distribution of satisficing planning in ensembles of random planning problems and in multiple runs of a randomized heuristic search on a single planning instance. Using common heuristic functions (such as FF) and six benchmark problem domains from the IPC, we find a heavy-tailed behavior, similar to that found in CSP and SAT. We investigate two notions of constrainedness, often used in the modeling of planning problems, and show that the heavy-tailed behavior tends to appear in relatively relaxed problems, where the required effort is, on average, low. Finally, we show that as with randomized restarts in CSP and SAT solving, recent search enhancements that incorporate randomness in the search process can help mitigate the effect of the heavy tail.

1 Introduction

The study of runtime distributions of several computational problems (most notably CSP and SAT) found heavy-tailed behavior for both ensembles of random problems and multiple runs of a randomized backtracking search on a single problem (Gomes 2003). This behavior accounted for the phenomenon of *exceptionally hard problems (ehps)* that appear as outliers in ensembles of easy problems and helped inspire search enhancements such as randomized restarts and algorithm portfolios (Gomes 2003).

In domain-dependent satisficing heuristic search, recent works found a connection between problem difficulty and problem constrainedness on ensembles of random problems (Cohen and Beck 2017b) and discovered the existence of exceptionally hard instances (Cohen and Beck 2017a). These results suggest that a heavy-tailed behavior, similar to the one observed for CSPs and SATs, might be observed for satisficing planning.

In this work, we focus on the heavy-tailed behavior in domain-independent satisficing planning and make the following contributions:

1. We show that fat- and heavy-tailed behavior can be observed on ensembles of random planning problems across different domains and heuristic functions.
2. We present a variant of greedy best-first search that introduces a limited amount of randomization in the search procedure and show that heavy-tailed behavior can be observed in multiple runs of the randomized search procedure on a single problem.
3. We demonstrate how different notions of constrainedness that are commonly used in the modeling of planning problems can lead to a fat- or heavy-tailed distribution of search effort.
4. We show that recent methods of non-greedy random exploration can help mitigate the heavy-tailed behavior in a similar manner to randomized restarts in CSPs.

2 Background

The study of full runtime distributions of algorithms over a problem set, rather than just the median or the mean, often provides useful information that can contribute to the design of better algorithms. Previous work found exceptionally hard instances in many kinds of computational problems (e.g., Gent and Walsh, 1994), that were attributed to a fat- or heavy-tailed behavior on ensembles of random problems in the easy region of the phase transition (Gomes, Selman, and Crato 1997). This behavior does not appear in ensembles of highly constrained instances, for which the median search effort is very high and all instances are uniformly hard. However, as we relax the problems, we move to a statistical regime in which the median effort is low, and the hardest instances are *exceptionally* hard. The runtime distribution in this regime is characterized by a fat- or heavy-tailed behavior, i.e., a slow decay of the tail of the survival function.

Later, Gomes et al. (2005) showed that heavy-tailed behavior can also be found in the runtime distribution of a randomized search procedure on a *single* instance, suggesting that the *ehps* can be easily solved by minor changes in the search procedure such as randomization. This result has motivated much work on eliminating the heavy-tailed behavior using randomized restarts, portfolios, etc. (Gomes 2003).

2.1 Fat- and Heavy-Tailed Distributions

The tail of a distribution is the very large (small) values of the distribution that determines the shape of its right (left) side (Verzani 2014). Fat- and heavy-tailed distributions have

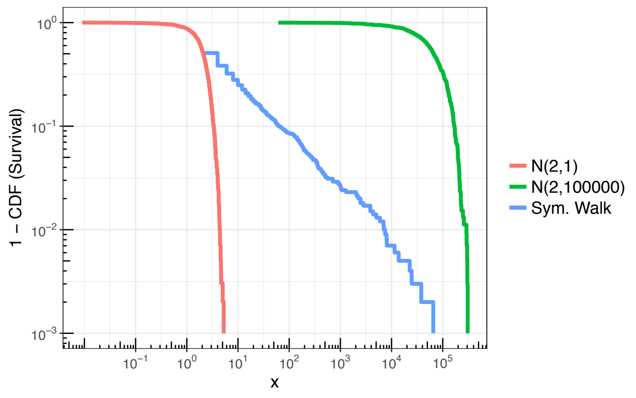


Figure 1: Heavy and non-heavy tailed behavior.

a long tail containing a considerable concentration of mass. Fat-tailedness can be determined based on the *kurtosis* of the distribution defined as $\kappa = \frac{\mu_4}{\mu_2^2}$, where μ_2 and μ_4 are the second and fourth moments, respectively.

Since moments are very sensitive to the most extreme points in the sample's tails and many heavy-tailed distributions do not even have asymptotically finite moments, we use the L-kurtosis measure. L-kurtosis, denoted as τ_4, is based on the L-moments (Hosking 1990) and can be thought of as a measure similar to kurtosis that gives less weight to the extreme tails of the distribution, and is less sensitive to small sampling biases (for a detailed discussion of L-moments and specifically L-kurtosis see Hosking, 1990). τ_4 is in the range $(-\frac{1}{4}, 1)$ and the normal distribution has $\tau_4 = 0.1226$. Distributions with higher value are called leptokurtic and are considered to be fat-tailed.

Heavy-tailed distributions are considered "heavier" than fat-tailed distributions, and all the moments of a heavy-tailed distribution are infinite above some order (Gomes et al. 2000). A random variable X with distribution function $F(x)$ is considered heavy-tailed if it has a Pareto-like decay of its tail above some threshold x_l (Resnick 2007), i.e., there exists some x_l>0, c>0, α>0 such that

$$1 - F(x) = P[X > x] = cx^{-\alpha}, x > x_l.$$

An approximately linear behavior over several orders of magnitude in the log-log plot of $1 - F(x)$ (i.e., the survival function) is a clear sign of heavy-tailed behavior with the slope providing an estimate of the stability index α (Gomes et al. 2000). In addition to the visual check, we can estimate α using the Hill estimator (Hill 1975) or by fitting a generalized Pareto distribution (GPD) model to the peaks over threshold (POT) using maximum likelihood (Smith 1987). If $1 < \alpha < 2$, X has infinite variance and if $0 < \alpha \leq 1$, X has both infinite mean and variance (Gomes et al. 2005). Due to the limitations of the Hill estimator (see Embrechts, Mikosch, and Klüppelberg, 1997), we use the POT method.

To demonstrate the difference between a heavy and a non-heavy tailed behavior, we use the example from Gomes et al. (2000). Figure 1 shows the log-log plot of $1 - F(x)$ vs. x for a normally distributed random variable with a mean of 2 and two possible values for the standard deviation. It also shows a random variable that represent the number of steps it takes for a symmetric random walk on a line to get back to its starting point. The normal distributions exhibit a fast-decay behavior, while the random walk exhibits a clear heavy-tailed behavior indicated by the approximately linear behavior on the log-log plot. The L-Kurtosis of both Normal random variables is $\tau_4 \approx 0.12$, while the random walk has $\tau_4 \approx 0.92$ and $\alpha \approx 0.63$.

From a practical point of view, the performance of a search algorithm on problem ensembles (or a repeated randomized search on a single problem) that exhibits either a fat- or a heavy-tailed behavior can benefit from randomization and restarts, with the heavier the tail, the greater the potential for a dramatic speed-up (Gomes 2003).

2.2 Randomized Search Algorithms

When analyzing the runtime distribution over an ensemble of problems, large variability (such as the one observed for *ehps*) can be either due to the variability among the instances in the ensemble or of the search algorithm itself. In order to isolate the latter, Gomes et al. (2000) studied the runtime distribution of multiple runs of a randomized algorithm over the same instance. A fat- or heavy-tailed behavior in the runtime distribution means that there is a low (but non-negligible) probability of very long runs and suggests that using randomized restarts can dramatically reduce the variability and potentially eliminate the heavy tail.

Gomes et al. (2000) proposed a method for adding randomization to complete, systematic, backtrack search algorithms such as DPLL. Traditionally, these algorithms construct a solution incrementally, and at each step a heuristic is used to decide how the partial solution will be extended (e.g., by assigning a value to a variable). Eventually, either a solution is found, or the algorithm backtracks to an earlier partial solution. If several decisions are deemed equally good, the algorithm typically applies some predefined rule to decide which decision to make. An easy way of introducing some randomization is to randomize the tie-breaking between decisions that are equally good, however, randomized tie-breaking might not be sufficient in many cases. Therefore, Gomes et al. (2000) introduced H, a "heuristic equivalence" parameter, that expands the set of decisions deemed equally good. Given this modification, each run of the algorithm on a specific instance will differ in the order of decisions, and potentially in the runtime.

2.3 Constrainedness of Problems

Different works on combinatorial search used different notions of constrainedness. In the analysis of random 3-SAT problems, Mitchell, Selman and Levesque (1992) used clause-to-variable ratio. For random CSPs, Smith and Dyer used the tightness of the constraint graph (1996). In random heuristic search problems, Cohen and Beck (2017b) used the density of edges in transition graph.

For structured benchmark domains, domain-specific parameters can be used. Examples of such parameters are the percent of pre-assigned colors in the quasigroup completion problem (Gomes et al. 2000) or the probability of a blocked cell in Grid Navigation problem (Cohen and Beck 2017a).

Gent et. al. (1996) suggested a unified definition of constrainedness that is based on the expected number of feasible solutions, however it is not easy to calculate this number for a planning problem (Cohen and Beck 2017a).

3 Analytical Framework

In this section we describe the analytical framework we use to investigate heavy-tailed behavior in planning. We first describe two notions of constrainedness of planning problems that we can manipulate in order to observe the different statistical regimes. Then, we propose a method to randomize a deterministic planning algorithm in order to analyze the runtime distribution on a single problem instance. Finally, we describe the benchmark problems we use.

3.1 Constrainedness of Planning Problems

A systematic study of fat- and heavy-tailed behaviors in satisficing planning involves analyzing the runtime distribution of instances of different constrainedness. In planning problems, there are various parameters that can effect the constrainedness of the problem (i.e., the expected number of feasible solutions). We describe two types of parameters that are often used to model planning problems.

Resource Constrainedness When planning with consumable resource, i.e., resources that cannot be replenished, resource constrainedness is the amount by which the initial resource supply exceeds the minimum need (Hoffmann et al. 2007; Gerevini, Saetti, and Serina 2008). The resource constrainedness can be measured by a parameter $C \geq 1$, namely the maximum number by which we can divide the resource supply without rendering the task unsolvable (Nakhost, Hoffmann, and Mueller 2012). Nakhost et al. studied the performance of state-of-the-art domain-independent planners as a function of C. To do so, they introduced an extended benchmark suite with three benchmark domains: *NoMystery, TPP, Rovers.*

Goal Constrainedness The definition of a goal condition in a planning problem can also be used to control the constrainedness of a problem. Consider a goal condition g_i, we use G_i to denote the induced set of goal states (i.e., states that satisfy the goal condition). We consider g_j to be a relaxation of g_i if $G_i \subseteq G_j$, meaning that every state that satisfies g_i will satisfy g_j. For example, if g_i is the conjuction of a set of propositions P and g_j is the conjuction of a set of propositions P' such that $P' \subseteq P$, we consider g_j to be a relaxation of g_i. There are, however, more nuanced relaxations that are not based on dropping goal propositions and we provide an example in our experiments.

3.2 Randomized Heuristic Search

In order to analyze the runtime distribution on a single instance, we need to introduce a limited amount of randomization into the search. In greedy best-first search (GBFS), the heuristic function determines the order of expansions. However, most commonly used heuristics are deterministic and so re-running the same instance will yield the same h-values for all states (notable exceptions are heuristics that are based on a random sample of the state space, e.g., Haslum et al., 2007). Although random tie-breaking can provide some randomness, in many domains it may not be sufficient.

We therefore present a general, parameterized method to randomize a heuristic function, and use it in our empirical analysis. Given a heuristic function $h(x)$ and a parameter $p \geq 0$, that represents the extent of randomization, we consider $h_{\Delta p}(x)$ to be an p-randomized version of h if for all states s: $h_{\Delta p}(s) = h(s) + \Delta_p^h$ where Δ_p^h is a random number in the range $[-p{\cdot}h(s), p{\cdot}h(s)]$. When used with deferred heuristic evaluation (Helmert 2006), we randomize the order in which the successors of a node are generated.

This method is similar to the one proposed by Gomes et. al. (2000) for backtracking search, as it randomly orders nodes that are within $100{\cdot}p$ percent of their h-values, and does not effect the completeness of the search algorithm.

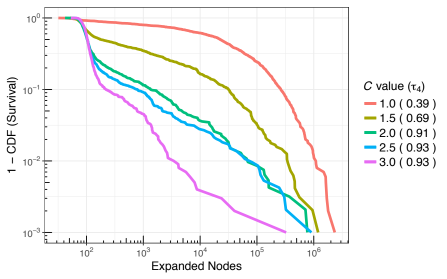
3.3 Benchmark Problems

To study the effect of resource constrainedness we use the benchmark domains used in Nakhost et. al. (2012): NoMystery, Rovers, and TPP. To study the effect of goal constrainedness we use the domains Maintenance, Parking, and Freecell. Following is a brief description of each domain.

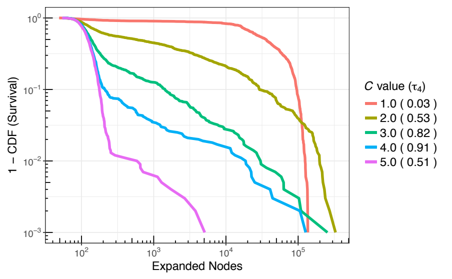
- NoMystery is based on the domain used in IPC'11. The goal is to transport packages between locations, using a set of trucks with a limited amount of fuel.
- Rovers is based on the domain from IPC'02, without the "recharge" operator (to make the resource consumable). The goal in this domain is to take a number of rock samples and images and transfer them to a lander.
- TPP was used in IPC'06. An agent is required to buy a set of items that are being sold at different prices in different markets. The limited resource is money, which is required for buying the items and for driving between markets.
- Maintenance was used in IPC'14. Mechanics work each day at one airport. Each plane visits some airports on given days. The goal is to schedule the mechanics' visits to the airports such that each plane will be maintained once.
- Parking (IPC'11, IPC'14) involves parking cars at N curbs where cars can be doubled-parked but not tripled-parked. The problem is to find a plan that moves from one configuration of cars to another by moving cars between curbs.
- Freecell (IPC'00, IPC'02) involves moving cards from initial configuration to a suit-sorted collection of stacks using the available free cells.

4 Empirical Analysis of Heavy-tailed Behavior in Satisficing Planning

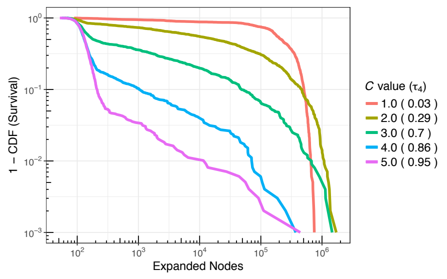
In this section we present an empirical analysis of the runtime distribution of the benchmark problems for different levels of constrainedness. We use GBFS and configure the planner not to re-open nodes. The heuristic function being used is FF with deferred heuristic evaluation. Experiments with standard heuristic evaluation yielded similar results and are omitted due to space.



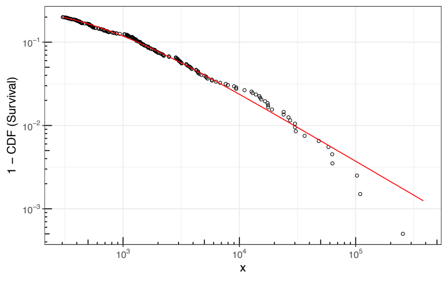
(a) NoMystery: 1000 random instances with FF.



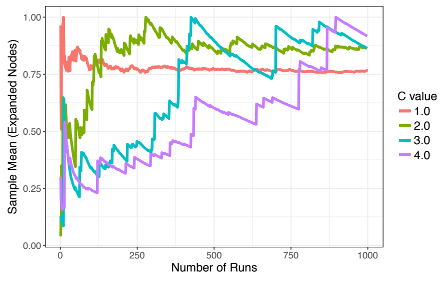
(b) NoMystery: 1000 randomized search runs on a single instance with FF.



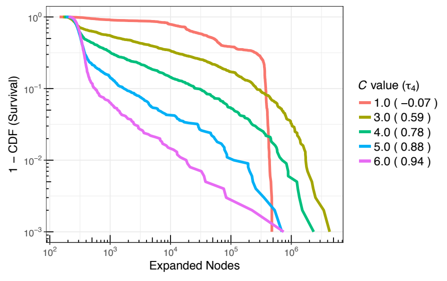
(c) NoMystery: 1000 randomized search runs on a single instance with landmark count.



(d) NoMystery: Fitting a GPD model with $\alpha \approx 0.82$ to the tail of $C = 3$.



(e) NoMystery: Stabilized vs. erratic behavior of the mean over increasing number of runs.



(f) Rovers: 1000 runs of a randomized heuristic search on the same instance.

Figure 2: Empirical results for resource constrainedness.

We start with a thorough analysis of the first domain (NoMystery) and present summarized results for the rest.

4.1 Resource Constrainedness

NoMystery Figure 2a shows the search effort distribution for 1000 random instances for different values of C, using h^{FF} (Hoffmann and Nebel 2001). For C=1, the tail decays much faster than the more relaxed problems. For C=2 we see a clear heavy-tailed behavior (a near-linear behavior over a several orders of magnitude). Although the problems in C=2 are, on average, much easier, the hardest instances are significantly harder than the median, compared to C=1.

More interesting is the search effort distribution of a randomized search algorithm on one instance. We used the median instance from the ensemble C=1, and created relaxed instances by increasing C. Figure 2b shows the search effort distribution of 1000 runs of a randomized search with $h^{FF}_{\Delta 0.05}$ on the same instance for different values of C. The results clearly show a transition from a statistical regime in which the problem is very constrained and the tail decays quickly (C=1) to a regime in which the problem is more relaxed and the tail decays much more slowly. For C=3.0, we see a clear heavy-tailed behavior with extremely long runs that are even longer than the longest run for C=1. The τ_4 values support this observation as C=1 has a lower τ_4 than the normal distribution indicating a very thin tail. As we increase C the τ_4 value increases, reaching 0.91 for C=4. As we relax the problem further we start see a decline in the heavy-tailed behavior and the corresponding smaller τ_4 value.

To estimate the stability index α of the tail, we fit a GPD model to the peaks over threshold using the maximum likelihood method (Carmona 2014). For C=3 we used a tail that corresponds to the largest 20% samples (chosen based on a visual inspection, see Figure 2b). We estimated $\alpha \approx 0.82$, and the quality of the fit is presented in Figure 2d. The estimated value suggests that the underlying distribution has an infinite mean and variance ($0 \leq \alpha \leq 1$).

Figure 2e shows the mean effort (normalized to a 0-1 scale) over an increasing number of runs for different values of C. When C=1, the mean stabilizes after a small number of samples (similar to the normal distribution). As we increase C, the mean takes longer to stabilize and still exhibits a large variance. When we move to the heavy-tailed regime, the mean does not stabilize with increasing sample size. This pattern is consistent with the erratic behavior of the mean, observed by Gomes et al. (2000) in CSPs. As we increase C further, problems gradually become universally easy and the mean starts to stabilize again (not presented).

To demonstrate that this phenomenon is not unique to h^{FF}, we also analyzed the runtime distribution using the landmark count heuristic (Richter, Helmert, and Westphal 2008), the landmark cut heuristic (Helmert and Domshlak 2009), and the causal graph heuristic (Helmert 2004), all with p=0.1. We present the result for landmark count (Figure 2c), and provide the τ_4 values for each C in Table 1. For all heuristics, we found a transition to a heavy-tailed regime as we relax the problem, however, the heavy-tailed regime occurred at a higher C value compared to h^{FF}(for landmark count at C=5.0, compared to C=4.0 for FF). This result is also consistent with CSPs, where the effect of using a less informative heuristic is that the heavy-tailed behavior shifts to a less constrained region (Gomes et al. 2005).

Domain	C/λ	τ_4	Domain	C/λ	τ_4
NoMyst (lmcut)	1.0	0.02	TPP	1.0	0.17
	2.0	0.42		1.5	0.58
	3.0	0.68		1.6	0.60
	4.0	0.85		1.8	0.43
NoMyst (CG)	1.0	0.06	Freecell	1.0	0.41
	2.0	0.37		2.0	0.73
	3.0	0.57		2.5	0.71
	4.0	0.70		3.0	0.45

Table 1: Summarized results for other domains.

Rovers We performed similar analysis for the Rover domain and found heavy-tailed behavior for both 1000 random instances and 1000 runs of a randomized search ($h^{FF}_{\Delta 0.05}$) on the median instance. Figure 2f shows the runtime distribution on a single instance. We can see that when C=1 we see a non-heavy tailed behavior, with kurtosis that is smaller than the normal distribution. As we increase C we see a fat-tailed behavior and finally a clear heavy-tailed behavior when C=5 and C=6, with $\alpha \approx 0.62$.

TPP For the TPP domain, as the problem generator is no longer available we only analyzed the runtime distribution of randomized search on single instances. Again, we observe a heavy-tailed behavior as we increase C. We omit the detailed results and provide the τ_4 values in Table 1.

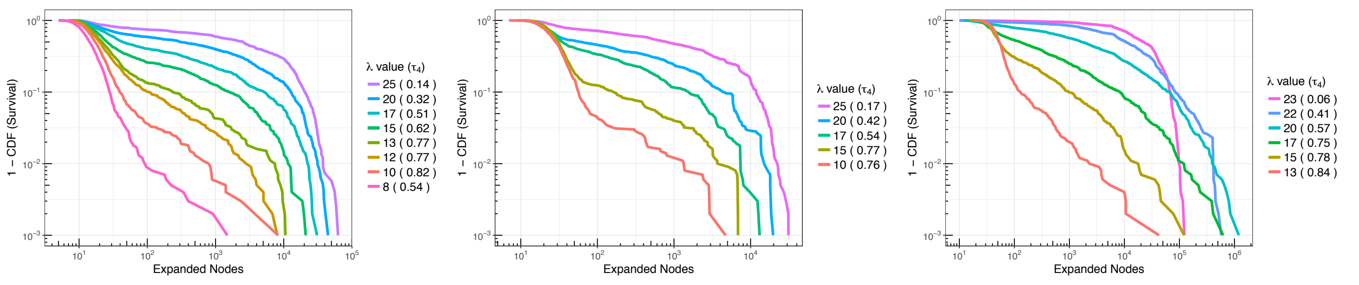
4.2 Goal Constrainedness

Maintenance We analyze the effect of the goal constrainedness by relaxing the goal of having all n airplanes checked by a mechanic. We start by using a simple, however a bit artificial, relaxation of the goal by requiring that only a chosen subset of λ airplanes be checked ($|\lambda| \leq n$). We consider a problem with 8 days and 25 planes, and analyze the runtime distribution for different λ sets. Figure 3a shows the runtime distribution of 1000 random instances and exhibits similar patterns to before. Figure 3b shows the distribution for a randomized search with $h^{FF}_{\Delta 0.1}$ on one instance. Again, we used the median instance of the more constrained ensemble ($|\lambda|$=25), however this time we had to manually create relaxed variants by picking a random subset of planes (we repeated the process several time and observed similar patterns). As we relax the problem we observe a fatter tail, and a truncated heavy-tail behavior (linear behavior in the log-log plot that is truncated). We observe increasing τ_4 from 0.17 (near-normal value) to 0.77. We estimated $\alpha \approx 0.51$, however the last few points diverged due to the truncated behavior.

We also analyze an alternative relaxation, of a more combinatorial nature: at least λ airplanes will be checked, but we do not decide which ones. Figure 3c shows the results for different λ values. Again, we see a fat- and heavy-tail behavior, but the tails are much fatter (τ_4 significantly higher), and we can observe a clear heavy-tailed behavior with $\alpha \approx 0.92$.

Parking We analyzed the effect of the goal constrainedness by allowing each car to park in more than one location. We extended the problem to support more than one goal configuration and create random instances of different goal constrainedness by introducing a parameter λ to control the number of allowed locations for a car. The runtime distribution on ensembles of random problems shows a heavy-tailed behavior as we increased λ, and is omitted due to space. Again we use the median instance and analyze the runtime distribution on a single instance for different λ values. The results (Figure 4) do not exhibit a Pareto-like heavy-tailed behavior, however they exhibit a clear fat-tailed behavior. For the more constrained instances (λ=$1, 1.5$), the τ_4 indicates a tail behavior that is similar to the normal distribution. For the more relaxed problems, we observe a very fat-tailed behavior and the corresponding high τ_4 values.

Freecell Originally, each card with face value i has to placed on top of a card with face value $i-1$ in each stack. We define a parameter λ that controls the number of cards that each card can be placed on top of, in each stack and manipulated existing instances from IPC'02 to create instances of different constrainedness. We omit the detailed results due to space but report the τ_4 values of 1000 runs of a randomized search with $h^{FF}_{\Delta 0.05}$ on one instance in Table 1. We found a clear heavy tailed behavior when λ=2.0, where the hardest instances (approximately 2%) were not solved in the 60 minutes time limit (we omitted them from the τ_4 calculation and so the reported value is smaller than the real one).



(a) Search effort distribution for 1000 random instances.

(b) 1000 runs of a randomized search on a single instance with FF.

(c) 1000 runs of a randomized search on a single instance with alternative relaxation.

Figure 3: Empirical results for the Maintenance domain.

5 The Impact of Randomized Exploration

In combinatorial search, the discovery of fat- and heavy-tailed behavior has inspired "boosted" search methods that employ randomized restarts in order to eliminate the heavy-tailed behavior, achieving significant speedups on hard real-world problems (Gomes, Selman, and Kautz 1998).

In satisficing planning, several works have suggested incorporating non-greedy *random exploration*, in which the search allocates limited time to expand nodes with non-minimal h-values. Two recent approaches are ϵ-GBFS (Valenzano et al. 2014) and Type-GBFS (Xie et al. 2014).

ϵ-GBFS expands a node selected uniformly at random from the open list with probability ϵ and the minimal h-value node with probability $1-\epsilon$. Empirical analysis showed that for ϵ values between 0.05 to 0.5, the number of domains for which the coverage increased was significantly higher than the number of domains for which the coverage decreased.

Type-GBFS utilizes type systems of heuristic search (Lelis, Zilles, and Holte 2013) to perform exploration. It maintains an additional queue that performs type-based exploration using a two level type bucket structure: first it picks a bucket b uniformly from all the buckets, then it picks a node n uniformly from all the nodes in b (Xie et al. 2014). Empirical analysis showed that Type-GBFS significantly outperforms standard GBFS, solving almost 200 more problems out of 2112.

In this section we examine the effect of these methods on the heavy tail phenomenon. Given the principled use of randomized restarts to escape heavy-tailed behavior in other combinatorial problems, an obvious question is whether the benefits of the random exploration approaches are higher in heavy-tailed regimes. Again, we start with a detailed analysis of the NoMystery domain and provide summarized results for the other domains.

NoMystery Figure 5 compares the effort distribution on a single instance for $h^{FF}_{\Delta 0.05}$ in a standard GBFS to an ϵ-GBFS with ϵ=0.2 and ϵ=0.5 (dotted and dashed lines, respectively). The results show an interesting pattern: for the more relaxed problems, that exhibited fat- or heavy-tailed behavior, ϵ-GBFS significantly reduces this behavior. Table 2 shows the τ_4 values for each C value for GBFS and ϵ-GBFS with ϵ=0.2 and ϵ=0.5. For C>2, ϵ-GBFS exhibits thinner tails for both values of ϵ. Note that for C=1 we see fatter tails and harder extreme instances as we increase ϵ. Furthermore, as Figure 5 shows, the thinner tails come at the cost of increased effort for the median case. Still, the most extreme cases are avoided, consistent with the increased coverage of this method.

Figure 6 compares the runtime distribution of a randomized FF heuristic in a standard GBFS (solid lines) to a type-GBFS (dashed lines). To directly address the impact of a non-greedy exploration, we use a simple type system that uses the h-values of the same heuristic rather then a more sophisticated one that uses new information. While the exact numerical results are different from ϵ-GBFS, the results show similar trends. The measured τ_4 values in Table 2 are also close to the ones observed for ϵ-GBFS with $\epsilon = 0.5$.

Figure 7 shows the behavior of the mean search effort over increasing number of samples. Consistent with eliminating the heavy-tailed behavior, we no longer observe the erratic behavior in Figure 2e. Instead, the mean stabilizes quickly for all C.

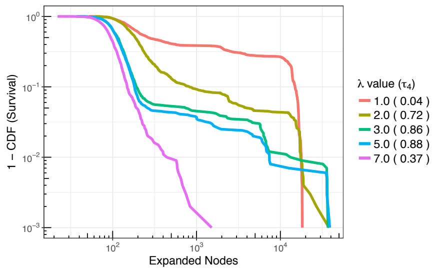


Figure 4: Parking: 1000 runs of a randomized search on a single instance.

C	*GBFS*	ϵ=.2	ϵ=.5	*Type*
1.0	0.03	0.16	0.25	0.17
2.0	0.53	0.42	0.32	0.35
3.0	0.82	0.68	0.55	0.58
4.0	0.91	0.67	0.62	0.60
5.0	0.51	0.37	0.35	0.31

Table 2: NoMystery: The effect of ϵ-GBFS and Type-GBFS on τ_4.

domain	C/λ	*GBFS*	ϵ=.2	*Type*
Rovers	4.0	0.78	0.62	0.54
TPP	1.6	0.51	0.37	0.31
Maint.	15	0.78	0.47	0.31
Parking	5.0	0.88	0.73	0.66
Freecell	2.0	0.73	0.43	0.43

Table 3: The effect of ϵ-GBFS and Type-GBFS on τ_4 for different domains.

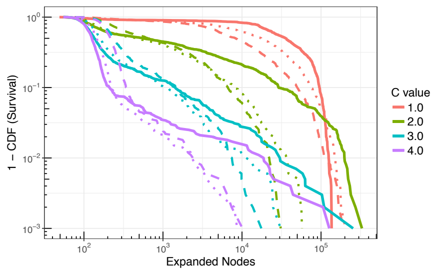


Figure 5: NoMystery: GBFS (solid) vs. ϵ-GBFS (ϵ=0.2: dotted, ϵ=0.5: dashed).

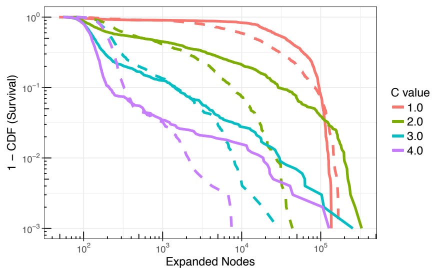


Figure 6: NoMystery: GBFS (solid) vs. Type-GBFS (dashed).

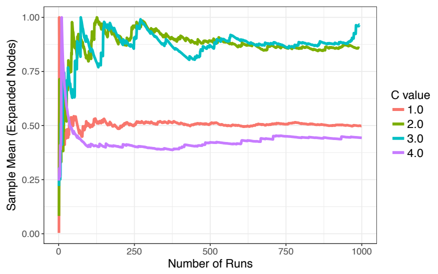


Figure 7: NoMystery: Stable behavior of the mean when using Type-GBFS.

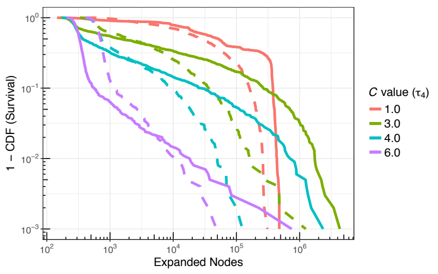


Figure 8: Rovers: GBFS (solid) vs. Type-GBFS (dashed).

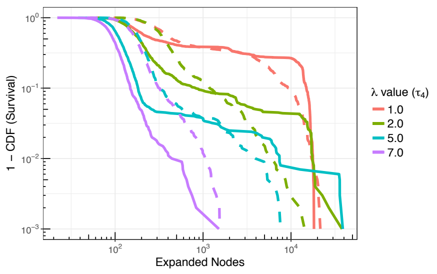


Figure 9: Parking: GBFS (solid) vs. Type-GBFS (dashed).

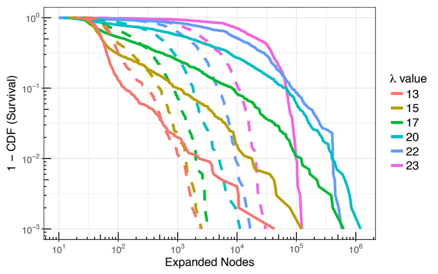


Figure 10: Maintenance: GBFS (solid) vs. Type-GBFS (dashed).

Rovers, TPP, Parking, Maintenance, Freecell Empirical analysis of these domains yielded similar trends. Figures 8, 9, and 10 compare standard GBFS with Type-GBFS for the domains Rovers, Parking, and Maintenance respectively. Table 3 shows the τ_4 values for one heavy-tailed constrainedness value for each domain for standard GBFS, Type-GBFS and ϵ-GBFS with $\epsilon = 0.2$. Notice that for the Maintenance domain, we also observe a significant improvement for the most constrained problems. Also, for the most relaxed problems in Parking, the random exploration makes the problems uniformly harder.

Our results show a clear pattern: incorporating elements of random exploration in the search helps to reduce and even eliminate the heavy tailed behavior. As previous works attributed the success of these methods to their ability to escape local minima (Valenzano et al. 2014; Xie et al. 2014), an obvious hypothesis is that relaxed problems tend to have few, but very large, local minima.

6 Discussion

The empirical results suggest that a heavy-tailed behavior can be observed for different planning problems, using different heuristic functions. This work provides a deeper understanding of the empirical difficulty of planning problems and accounts for the previously discovered *ehps*. We also provide evidence that the use of random exploration in the search procedure addresses the heavy-tailed behavior, in a manner similar to the randomized restarts in CSPs.

Heavy-tailed behavior has been shown not to be inherent to backtracking search in general, but rather to depend on the efficiency of the search procedure as well as on the level of constrainedness of the problem (Gomes et al. 2005). In the context of planning, uninformative heuristics will not exhibit a heavy-tailed behavior, even as we relax the problems.

However, similar to combinatorial search, we find that, in practice, heavy-tailed behavior can be observed for many problems using common heuristics. Specifically, resource constrained problems seem to exhibit such behavior when solved with a heuristic that is based on ignoring the delete effects. For highly constrained problems, most paths do not lead to a solution due to lack of resources. As we relax the problems, we can usually find a solution easily, however one mistake can still lead to the need to exhaust a region of the state space that has no solution, resulting in an extremely long run. We also show that heavy-tailed behavior can be observed as we relax highly constrained problems by introducing more goal states. In the relatively goal-relaxed problems, while most paths will lead to a goal state, one heuristic mistake can lead the search into a region with no solution, e.g., when achieving one goal proposition has a delete effect (that the heuristic does not account for) that prevents us from achieving another.

In combinatorial search, heavy-tailed behavior has been shown to be correlated with an exponential distribution of depths of the subtrees with no solutions (Gomes et al. 2005). In satisficing planning, local minima (i.e., regions of the state space that have no solution; Wilt and Ruml, 2014) have been shown to have a negative effect on the search effort (e.g., Xie, Müller, and Holte, 2015), and the results in Section 5 lead us to hypothesize that the instances in the heavy-tailed regimes will exhibit a small number of very large local minima. Investigating the correlation between the observed heavy-tailed behavior and the distribution of the sizes of local minima is a future direction of our research.

7 Conclusion

We performed an empirical analysis of the runtime distribution on ensembles of random planning problems and multiple runs of a randomized search procedure on a single planning problem. We considered two notions of constrainedness in planning problems and showed that relaxed problems often exhibit fat- or heavy-tailed behavior.

Our empirical analysis provides evidence that the previous results on the existence of exceptionally hard problems (Cohen and Beck 2017a) and the success of randomized exploration (Valenzano et al. 2014; Xie et al. 2014) are due to a fat- or heavy-tailed behavior. Our results also connect planning to a body of literature on other computational problems for which similar results have been obtained.

We expect our results to help in the design of more informed planning algorithms that specifically address heavy-tailed behavior. Furthermore, our results suggest that algorithm benchmarking should consider instances from across the constrainedness range and sample multiple random instances at each level of constrainedness.

Acknowledgements We thank the anonymous reviewers whose valuable feedback helped improve the final paper. The authors gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada.

References

Carmona, R. 2014. *Statistical analysis of financial data in R*. Springer, second edition.

Cohen, E., and Beck, J. C. 2017a. Cost-based heuristics and node re-expansions across the phase transition. In *International Symposium on Combinatorial Search (SoCS)*, 11–19.

Cohen, E., and Beck, J. C. 2017b. Problem difficulty and the phase transition in heuristic search. In *AAAI Conference on Artificial Intelligence (AAAI)*, 780–786.

Embrechts, P.; Mikosch, T.; and Klüppelberg, C. 1997. *Modelling Extremal Events: For Insurance and Finance*. Springer-Verlag.

Gent, I. P., and Walsh, T. 1994. Easy problems are sometimes hard. *Artificial Intelligence* 70(1):335–345.

Gent, I. P.; MacIntyre, E.; Prosser, P.; and Walsh, T. 1996. The constrainedness of search. In *National Conference on Artificial Intelligence (AAAI)*, 246–252.

Gerevini, A. E.; Saetti, A.; and Serina, I. 2008. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence* 172(8-9):899–944.

Gomes, C. P.; Selman, B.; Crato, N.; and Kautz, H. 2000. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of automated reasoning* 24(1):67–100.

Gomes, C. P.; Fernández, C.; Selman, B.; and Bessière, C. 2005. Statistical regimes across constrainedness regions. *Constraints* 10(4):317–337.

Gomes, C. P.; Selman, B.; and Crato, N. 1997. Heavy-tailed distributions in combinatorial search. In *International Conference on Principles and Practice of Constraint Programming*, 121–135. Springer.

Gomes, C. P.; Selman, B.; and Kautz, H. 1998. Boosting combinatorial search through randomization. *National Conference on Artificial Intelligence (AAAI)* 431–437.

Gomes, C. 2003. Randomized backtrack search. In Milano, M., ed., *Constraint and integer programming: Toward a unified methodology*. Springer Science & Business Media. 233–291.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *National Conference on Artificial Intelligence (AAAI)*, 1007–1012.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: what's the difference anyway? In *International Conference on Automated Planning and Scheduling (ICAPS)*, 162–169.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 161–170.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hill, B. M. 1975. A simple general approach to inference about the tail of a distribution. *The annals of statistics* 3(5):1163–1174.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J.; Gomes, C. P.; Selman, B.; and Kautz, H. A. 2007. SAT encodings of state-space reachability problems in numeric domains. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1918–1923.

Hosking, J. R. M. 1990. L-moments: Analysis and estimation of distributions using linear combinations of order statistics. *Journal of the Royal Statistical Society. Series B (Methodological)* 52(1):105–124.

Lelis, L. H.; Zilles, S.; and Holte, R. C. 2013. Stratified tree search: a novel suboptimal heuristic search algorithm. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 555–562.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problems. In *National Conference on Artificial Intelligence (AAAI)*, 459–465.

Nakhost, H.; Hoffmann, J.; and Mueller, M. 2012. Resource-constrained planning: A monte carlo random walk approach. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 181–189.

Resnick, S. I. 2007. *Heavy-tail phenomena: probabilistic and statistical modeling*. Springer Science & Business Media.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *AAAI Conference on Artificial Intelligence (AAAI)*, 975–982.

Smith, B. M., and Dyer, M. E. 1996. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence* 81(1):155–181.

Smith, R. L. 1987. Estimating tails of probability distributions. *The annals of Statistics* 1174–1207.

Valenzano, R. A.; Sturtevant, N. R.; Schaeffer, J.; and Xie, F. 2014. A comparison of knowledge-based GBFS enhancements and knowledge-free exploration. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 375–379.

Verzani, J. 2014. *Using R for introductory statistics*. CRC Press.

Wilt, C. M., and Ruml, W. 2014. Speedy versus greedy search. In *Symposium on Combinatorial Search (SoCS)*, 184–192.

Xie, F.; Müller, M.; Holte, R.; and Imai, T. 2014. Type-based exploration with multiple search queues for satisficing planning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2395–2402.

Xie, F.; Müller, M.; and Holte, R. 2015. Understanding and improving local exploration for GBFS. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 244–248.