On the Fine Structure of Large Search Spaces

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

Abstract

Recently there has been significant progress in our understanding of the computational nature of combinatorial problems. Randomized search methods, both complete and incomplete, often outperform deterministic strategies. In this paper, we relate the performance of randomized methods to geometric properties of the underlying search space. In particular, our study reveals the inherent fractal nature of the search space, at different length scales, for a range of combinatorial problems. We also discuss the impact of these results on the design of better search methods.

1 Introduction

In recent years, much progress has been made in terms of our understanding of the nature of search problems and the performance of search algorithms.

Many results have been obtained first through careful empirical studies, and later confirmed by a formal analysis. A key recent finding was the discovery of hard problem instances at phase boundaries, as they occur in combinatorial problems. (See [11] for a series of papers in this area.) Other results concern the study of the performance of search algorithms on a variety of benchmark problems.

Typically, one characterizes the perfomance of algorithms by considering the mean or the median runtime on a range of benchmark problems. Recently, more sophisticated approaches encompass the study of the full runtime distribution profiles, either on an ensemble of problem instances or on a series of runs of a randomized method on a single instance (see *e.g.*, [5], [9], and [12]). Performance is generally measured in terms of the number of backtracks, or another runtime related measure, for finding a solution or proving infeasibility. The characterization of search methods through the study of their runtime distributions has proven very insightful.

A concrete example of the impact of the study of runtime distributions involves local search methods. Hoos [12] shows that the runtime distribution of certain local search methods is exponentially distributed. As a consequence, as discussed in [12], one can obtain an optimal speedup on parallel runs. For complete methods, it has been observed that extreme variations in runtime can occur ([6], [7], [19] and [11]). In [9], Gomes *et al.* show that the extreme fluctuations of backtrack style procedures are the result of the so-called heavy-tailed nature of the underlying runtime distributions. This phenomenon can be exploited in algorithmic sense by using a rapid-restart strategy to speed up combinatorial search. Some examples of concrete speedups have since been found in planning and scheduling (*e.g*, [20], [10], and [21]).

Runtime distributions are obtained by solving an ensemble of problems or by repeated runs on a single instance. One obvious limitation of runtime distributions is to deal with an unsolved problem instance from a new problem domain. In this case, one would like to tune the search algorithms but no apriori information exists about the underlying runtime distribution.

We propose an approach to overcome this difficulty by studying the geometric properties of the search space. In our approach, no complete runs are needed to obtain a runtime distribution, instead one can use the information from a single, partial run. More specifically, we show how extreme variations in runtime distributions (heavy-tailedness) seem to correspond to a highly irregular search space. In contrast, our experiments also indicate that, when the runtime distribution is not heavy-tailed, the underlying search space is highly regular. We can measure the degree of irregularity of a search space by considering the fractal dimension of the fringe of the search tree. Using this approach, based on a relatively short partial run, one can determine the shape of the fringe of the search space, and therefore decide what type of search strategy (*e.g.*, rapid restarts) will be most effective.

2. Characterizing search methods: cost distributions

Backtrack style search methods can exhibit dramatic variations in solution time on different problem instances or, for randomized methods, even when running on the same problem instance with different random seeds. Gomes *et al.* [9] characterized these runtime distributions using non-standard probability distributions. These distributions are called heavy-tailed distributions, and were introduced to model phenomena that are characterized by extreme variations ([14], ([15] and [18]).

Figure 1 illustrates the heavy-tailed phenomenon. We consider a randomized backtrack search with a cutoff parameter, which gives the number of backtracks at which the search is terminated. We run this procedue for a range of cutoff values. The figure gives the fraction of unsuccessful runs as a function of the cutoff value. For example, consider the problem instance called "rand 500 / 3.5". This is a underconstrained, randomly generated 3CNF formula (satisfiable) with 500 variables and a ratio of clauses to variables of 3.5. At a cutoff of 1000 backtracks, ≈10% of the runs were unsuccessful; at a cutoff of 10,000, the fraction is ≈1%.[1] The key point to observe is that the fraction of unsuccessful runs decreases relatively slowly as a function of the cutoff. In fact, the distribution ranges over several orders of magnitude. (Note the log scale.) This is typical of heavy-tailed behavior. Formally, one observes a powerlaw decay in the underlying probability distribution. On a log-log plot this corresponds to (approximate) linear behavior of the tail of the distribution. We see in the figure that "rand500" and "logistic c" (a planning problem) exhibit heavy-tailed behavior. In contrast, "parity 16" and "rand 200 / 4.3" exhibit a sharp dropoff in the fraction of unsuccessful runs. In fact, these two instances are not heavy-tailed. The "parity 16" problem is a propositional encoding of a problem from cryptography [2]. The "rand200 / 4.3" is a critically constrained 3CNF formula with 200 variables.

The different underlying distributions give rise to different search strategies. For the non-heavy tailed instances, the optimal strategy is to simply run with a cutoff slightly above the point where the curves drop off sharply. From the figure, it is clear that on a single run, one will almost always solve the problem, assuming a cutoff value of around 4000 backtracks. The situation is quite different in the heavy-tailed case. A so-called rapid restart strategy is often most effective. In a sense, one should try to avoid the long tail of the distribution.

Figure 2 illustrates such a strategy. The figure shows the average number of backtracks performed until a solution is found for a large planning problem instance as a function of

[1]For our experiments, we used a randomized version of a state-of-the-art SAT solver, called satz [13].

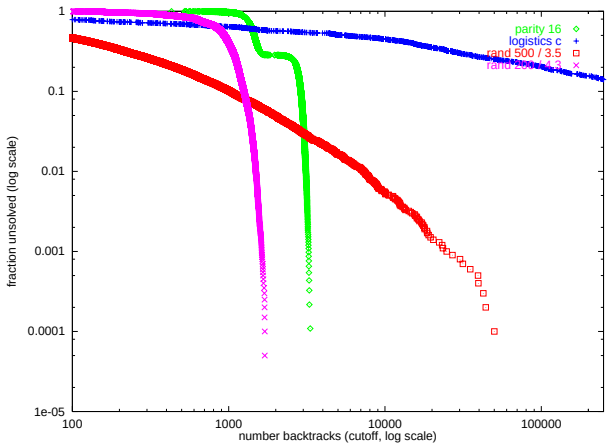

Figure 1. Runtime profiles of randomized complete search methods. Two examples of heavy-tailed behavior and two of non-heavy-tailed distributions.

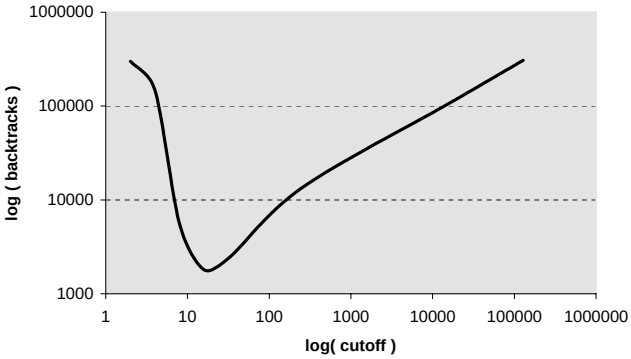

Figure 2. Illustration of rapid restart strategy.

different values of the cutoff (from 2 to 128,000). We see that a cutoff of around 20 leads to a minimal average solution time (around 2000 backtracks). At this cutoff, approximately 100 restarts are performed per solution. In contrast, at a higher cutoff value, *e.g.*, 100,000, roughly 1 in 3 runs is successful, and therefore the average total time per solution is around 300,000. The figure provides a good illustration of the effectiveness of a rapid restart strategy in the presence of heavy-tailed behavior.

3. Characterizing search spaces: a geometric approach

As we noted above, the main limitation of using runtime distributions to optimize search strategies is the need to

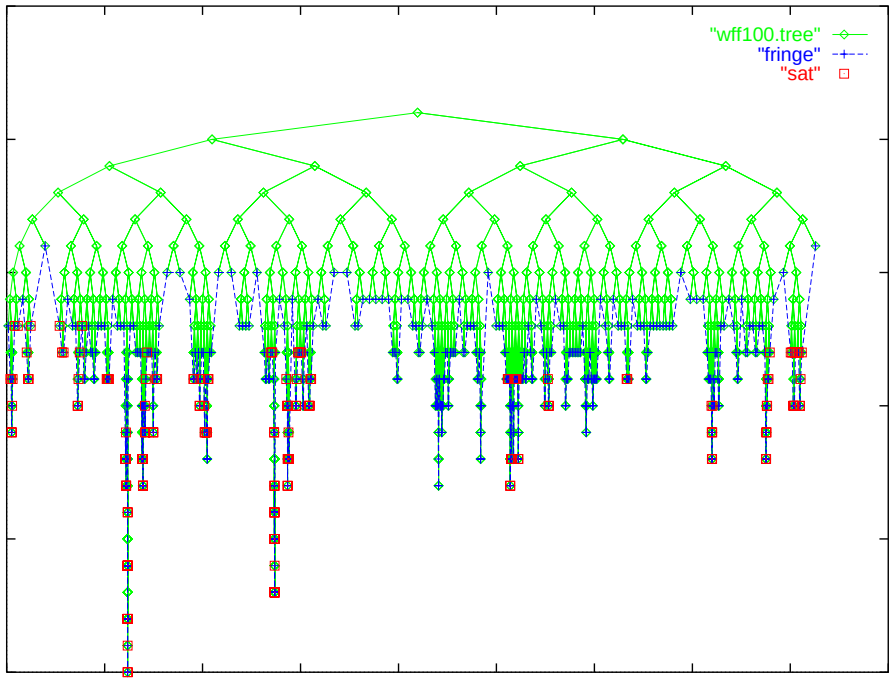

Figure 3. Example of backtrack tree.

collect detailed runtime data. This approach requires many runs on a single problem instance or on a collection of instances. Furthermore, the instances cannot be too difficult because we need to solve them repeatedly to collect reasonable statistics (typically 10,000 runs are needed).

We will now consider a different approach to determine the appropriate search strategy by studying the shape of the search tree, which can be estimated from a single, partial run on a single instance.

Figure 3 shows an example of a backtrack search tree. This tree was obtained by running a randomized Davis-Putnam search procedure [13] on a 100 variable, random 3CNF formula. In the plot, we have connected the leaves of the tree. We refer to this curve as the "fringe" of the tree. We also indicate leaf nodes that correspond to satisfying assignments.

Our hypothesis is that the shape of the tree, as characterized by its fringe, can be used to predict properties of runtime distributions, such as, its heavy-tailedness. Intuitively, a highly irregular fringe would explain heavy-tailed behavior. The solutions are distributed throughout the fringe of the tree, but how quickly a backtrack search procedure will find a solution will be highly dependent on where in the tree the search starts. In particular, a search may hit upon a large subtree without any solution, resulting in a large runtime.

In order to measure the degree of irregularity of the fringe of the tree, we consider its fractal dimension. The fractal dimension measures the "degree of meandering" or "roughness" of a geometric object. It is difficult to give a comprehensive introduction to fractals in this short paper. We will therefore only give a flavor of the ideas behind fractals and refer to the literature for more details (see *e.g.*, [16]).

The classical example used to introduce the notion of fractal dimension is the question of how to measure the length of the coastline of Great Britain. In the beginning of this century the scientist Lewis Richardson pointed out that the length of a coastline depends on the yardstick used to measure it: the smaller the yardstick, the longer the length measured, because more and more details of the coastline are taken into account. Figure 4 illustrates this phenomenon. When using a yardstick of length "S=1", the perimeter of the curve is 6. If we use a yardstick of length "S=2", the length of the curve is less than 6, and even smaller when the length of the yardstick is "S=3".

In fact, Richardson observed that when plotting the length of the coastline as a function of the precision of the measuring tool on a log-log plot, one obtains a linear relationship. Formally, we have $L = c(1/s)^d$, where L is the measured length, s is the length of the yardstick, and c and d are constants. Richardson obtained $d = 0.22$ for the coastline of Britain. Mandelbrot introduced the fractal dimension, D, defined as $d + 1$. A straight line has $D = 1.0$. A fractal dimension of 1.22, 0.22 above 1.0, indicates the

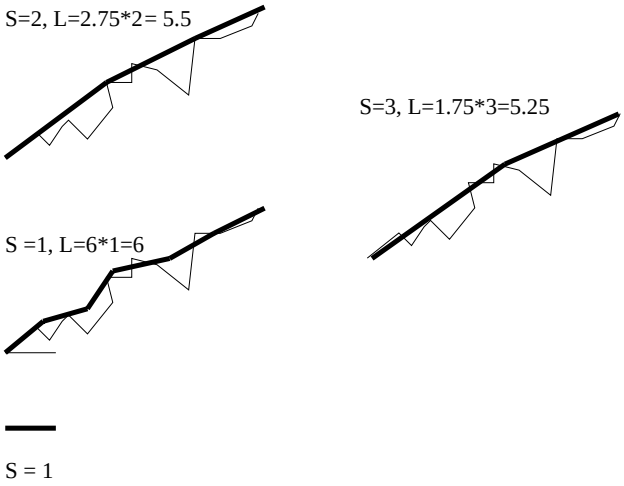

Figure 4. Using different yardsticks to measure the length of a curve.

"degree of meandering" or "roughness" of the curve.

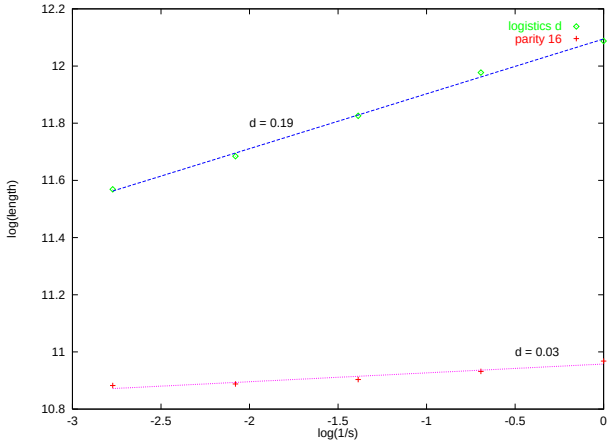

Figure 5. The length of the fringe of the search tree measured over a range of scales (s) for the logistics.d problem and the parity 16 instance.

We now consider the fractal dimension of the fringe of our search trees. (Note that the fringe is the curve obtained by connecting the leaves of the search tree.) Figure 5 shows the log-log plot of the length of the fringe as a function of the measurement length scale. The figure shows the results for two different search problems. The top set of data points corresponds the search tree of a large logistic planning problem (same instance as used in figure 2). The plot reveals a linear relationship. We have fitted a line using a least square fit. The angle of the line is 0.19. We thus obtain a fractal dimension of 1.19 for the fringe. For comparison, the bottom set of data points corresponds to the fringe of a search tree for the parity 16 problem (not heavy-tailed). The fitted line has a slope of 0.03, i.e., the fringe has a fractal dimension of 1.03, close to 1.

In other words, the fractal analysis shows that the fringe of the search tree for parity problem is much more regular than that for the logistics planning problem. (A more regular fringe means that all branches have roughly the same length. The fringe of a complete binary tree has $D = 1.0$.)

The fractal dimensions nicely correlate with the fact that the search on the logistic planning problem is strongly heavy-tailed whereas the distribution for the parity 16 problem is not.

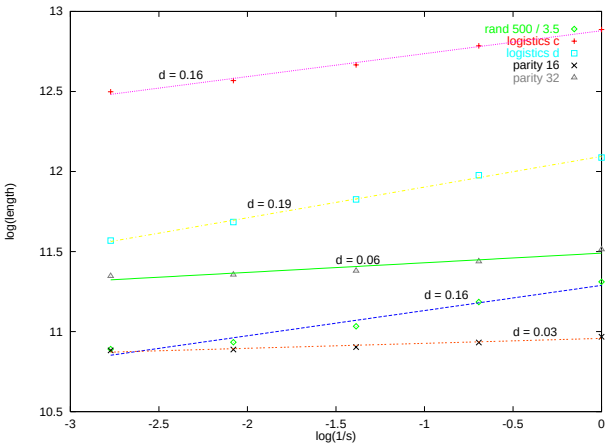

Figure 6. The length of the search tree fringe for a range of problems.

To obtain further evidence for the relation between fractal dimension and heavy-tailedness, we consider several other search problems. Figure 6 shows the results. (We included our two previous problem instances.) The three sloped lines ($d = 0.16, 0.19, 0.16$) correspond to heavy-tailed distributions, whereas the two near-horizontal lines ($d = 0.03, 0.06$) correspond to two non-heavy tailed problem instances (parity problems). The lines are not completely horizontal because the branches of the search tree are not exactly the same length, they fluctuate slightly. However, apparently not enough to cause heavy-tailed behavior. A direct visual inspection of part of the search tree confirms that the trees for rand500, logistics.c, and logistics.d are indeed much more irregular than the trees for the parity problem, as one would expect based on their fractal dimensions. Note that for this analysis, we only used a partial run (about 100,000 branches) of the search algorithm on each instance. The time on SUN workstation to obtain the trees was about 10 minutes on the easiest problem (parity 16) and 2hrs on the hardest (logistics.d). On these runs,

logistics d and parity 32 were not actually solved. This should be contrasted with the effort to create runtime distributions for these problems, as shown in figure 1. The runtime data for the heavy-tailed curves took many days to generate; also, it is not practically feasible to create cost distributions for parity 32 and logistics d. These instances are too difficult for obtaining a good sample of the distribution.

4. Conclusions

We have presented an alternative way to gain insight into combinatorial search spaces. In particular, we showed that the fractal dimension of the fringe of backtrack search tree can reveal whether the search space leads to heavy-tailed behavior. The main advantage of our approach is that it eliminates the need to obtain a full runtime distribution before deciding on a search strategy, such as rapid restarts. Our results are still preliminary. Further research is required.

An interesting future direction is to study the relation between the performance of various sophisticated backtrack strategies, such as dynamic backtracking and other lookahead and lookback strategies ([1], [3], [8], [4], and [17]). One conjecture is that a more sophisticated strategy will lead to a more regular search space with a lower fractal dimension. We hope that our approach will lead to further studies of the geometric properties of search spaces, enabling the design of better search methods.

Acknowledgments

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

References

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

[2] J. Crawford. Encoding parity problems. DIMACS Benchmark problem set, 1994.

[3] R. Dechter. Constraint networks. In *Encyclopedia of Artificial Intelligence*, 1991.

[4] E. Freuder and A. Mackworth. *Constraint-based reasoning*. MIT Press, 1994.

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

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

[7] I. Gent and T. Walsh. The Satisfiability Constraint Gap. *Artificial Intelligence*, 81, 1996.

[8] M. L. Ginsberg. Dynamic Backtracking. *Journal of Artificial Intelligence Reserach*, 1:25–46, 1993.

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

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

[11] T. Hogg, B. Huberman, and C. Williams. Phase Transitions and Complexity. *Artificial Intelligence*, 81, 1996.

[12] H. Hoos. PhD Thesis,TU Darmstadt, 1998.

[13] C. M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[14] B. Mandelbrot. The Pareto-Lévy law and the distribution of income. *International Economic Review*, (1):79—106, 1960.

[15] B. Mandelbrot. *The fractal geometry of nature*. Freeman: New York, 1983.

[16] H. Peitgen, H. Jurgens, and D. Saupe. *Chaos and Fractals: New Frontiers of Science*. Springer-Verlag: Berlin, 1992.

[17] P. Prosser. Forward checking with backmarking. In M. Meyer, editor, *Constraint Processing, LNCS 923*. Spring-Verlag, 1995.

[18] G. Samorodnitsky and M. Taqqu. *Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance*. Chapman and Hall, 1994.

[19] B. Smith and M. Dyer. Locating the Phase Transition in Binary Constraint Satisfaction Problems. *Artificial Intelligence*, 81, 1996.

[20] Vandengriend and Culberson. The G_{nm} Phase Transition is Not Hard for the Hamiltonian Cycle Problem. *Journal of Artificial Intelligence Reserach*, 1999.

[21] T. Walsh. Search in a Small World. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.