Is there a Constrainedness Knife-edge?

John Slaney[1]

Abstract. Recent work on search has identified an intriguing feature dubbed the *constrainedness knife-edge* by Walsh (Proc. AAAI-98, 406–411) whereby overconstrained problems seem to become on average even more constrained as the search goes deeper, while underconstrained ones become less constrained. The present paper shows that while the knife-edge phenomenon itself is real, many of the claims that have been made about it are incorrect. It is *not* always associated with criticality, it is *not* a function of the problem so much as of the algorithm used to solve it, and it does *not* help to explain the peculiar hardness of problem instances near phase transitions. Despite the negative findings, the upshot is that Walsh's work has opened a fascinating line of research which will surely repay further investigation.

1 Introduction

This paper is a critical examination of ideas put forward in [3] concerning the way in which the constrainedness of combinatorial search problems changes with the depth of the search. Briefly, [3] reports experiments showing a remarkable regularity in the relationship between constraint tightness and depth: overconstrained problems tend to become even *more* constrained as the search goes deeper in the tree, while underconstrained ones seem to become *less* constrained. In between is the "knife-edge", a region in which critically constrained problems (i.e. those at the solvability phase transition) remain critically constrained, independently of the depth at which one examines the tree. It is suggested in [3] that this may explain why problems near the phase transition are hard.

The results in the present paper are largely negative. It is shown here that:

1. The presence of a knife-edge is not inevitable: its position and even its existence depend on the search algorithm and heuristics as much as on constrainedness.
2. It does little or nothing to explain cost peaks associated with phase transitions in solvability.
3. It is somehow related to the effectiveness of search heuristics, but in detail this relationship is still obscure.

All of this leaves the knife-edge phenomenon rather enigmatic: it is obvious *that* it is interesting but far from obvious *why*.

In outline, the paper is organised as follows. §2 fills in some of the background, setting out the generic definition of constrainedness κ given in [1] and attempting to explicate it in terms of information. §3 is also mainly expository, recapitulating the discoveries reported in [3].

In §4 the notion of constrainedness changing with depth is developed further, resulting in a simple relationship between constrainedness *to* a given depth (that is, in the top part of the tree), constrainedness *from* that depth to the bottom of the tree, and the problem constrainedness κ. From this is easily obtained an expression for the derivative of constrainedness with respect to depth, from which it appears that the knife-edge need not coincide with the critically constrained region.

In §5 an experimental investigation casts more doubt on the idea that the knife-edge indicates criticality, and indicates that the phenomenon is, despite appearances, not part of the explanation of the familiar cost peak associated with the phase transition in solvability. The final section, §6, takes up another suggestion, that the shape and location of the knife-edge for a given search algorithm and given heuristics could be used as an indicator of how closely they approach the perfect solver for the problem. The conclusion is cautiously optimistic, since experimental results for 3-SAT are encouraging. However, the situation is not simple, as is shown by the example of the pure literal deletion rule for SAT, which greatly improves the efficiency of the Davis-Putnam algorithm but leaves the knife-edge less clear than it was before.

Throughout this paper, the example used for experiments is fixed clause length random 3-SAT. This is the standard illustration of the phenomenon and is the main example used in [3]. Although logically one example is enough to refute an hypothesis, clearly it would be of interest to examine other domains in the same way.

2 Constrainedness

A (finite) search problem consists of a (finite) set of *states* partitioned into those satisfying a condition or *constraint* and the rest. A natural way to define the probability of a constraint in the context of a state space is as the proportion of states which satisfy it. If a state is determined by the values of n independent bits or boolean variables, then there are 2^n possible states and the ones which satisfy the constraint may be picked out by a boolean formula in n variables which is true on exactly those assignments of values to its variables which correspond to the satisfying states. The probability of a boolean formula, on this view, is the proportion of truth value assignments on which it holds.[2]

Lozinskii [2], borrowing from standard information theory, defines the information content $\mathcal{I}$ of a propositional formula as the negative logarithm of its probability in the above sense. That is, where the variables in the formula are $p_1, \ldots, p_n$ and where M is the number of assignments of truth values to these variables on which the formula

[1] Automated Reasoning Project, Australian National University, Canberra, ACT 0200, Australia. e-mail: John.Slaney@anu.edu.au

[2] This somewhat simplistic idea has deep roots—tracable back to Laplace. The reformulation in terms of boolean combinations of bits is also ancient, being central to the early twentieth century philosophy of logical atomism [4],§5.15. In computation theory it has recently become popular again—at least as a useful fiction.

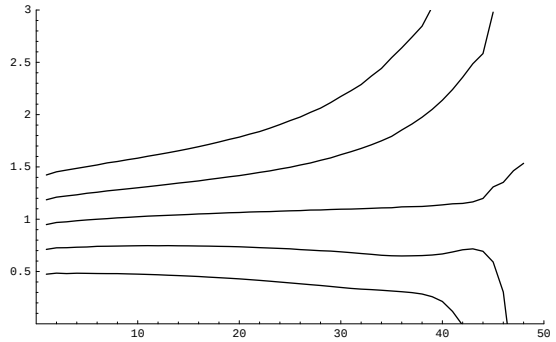

Figure 1. κ_i plotted against depth: Davis-Putnam algorithm (DP) for fixed clause length 3-SAT with 50 variables.

is true:

$$\mathcal{I} \quad = \quad n - log_2 M$$

More recently, Gent, MacIntyre, Prosser and Walsh [1] have sought to provide a general account of how constrained an ensemble of search problems is, devising a quantity κ which amounts roughly, in Lozinskii's terms, to $\mathcal{I}/n$, the average information content per variable:

$$\kappa = 1 - \frac{\log_2 \langle sol \rangle}{n}$$

Here $\langle sol \rangle$ is the expected number of solutions to a problem in the ensemble, and n is the logarithm (base 2) of the size of the state space. It is important to note that this definition applies to sets rather than to single problems, and that it is not parochial to boolean satisfiability but makes sense for search problems in general. An important effect of defining κ for sets of (cognate) problems is that it can have a finite value even where some of the problems in the set have no solutions.

Evidently, $\kappa = 0$ if $\langle sol \rangle = 2^n$: that is, if every state is a solution. This is intuitively right, for in such a case the problems are completely unconstrained. At the other extreme, κ tends to infinity (the problems become infinitely constrained) as the average number of solutions tends to zero—as the constraints become so strong as to rule out all solutions. In between, an important threshold occurs at $\kappa = 1$ where the problems are critically constrained, with an average of one solution each.

If an average of $\langle sol \rangle$ solutions per problem instance were scattered over large sets of instances randomly and independently, with uniform distribution, then a phase transition in solvability would appear exactly where $\kappa = 1$: instances in sets with $\kappa < 1$ would almost certainly have solutions, while those in sets with $\kappa > 1$ would almost certainly not. If the problems were completely unstructured, so that a solution on one branch of a search tree was independent of whether there is a solution on any other branch, then a peak in computational cost would be associated with such a phase transition, not because critically constrained problems have special features but just because the hardest searches are for scarce needles in large haystacks. Real problems are not exactly like that: solutions are not independent, and there is structure in each problem instance. However, a wide range of real problems are *enough* like that to show solvability phase transitions near $\kappa = 1$ and corresponding cost peaks. Hence one reason why κ is important is that it provides a unified account of phase transition phenomena in combinatorial search problems.

3 The constrainedness knife-edge

The title of this section echoes that of a fascinating paper by Walsh [3] in which the question is raised of how the (apparent) constrainedness of search problems changes with depth in the search tree. The paradigm example of such a search problem is k-SAT, for which the simplest definition of constraint tightness yields an estimate of κ:

$$\kappa = \frac{-\sum_{j=1}^{k} \mathcal{C}_j \log_2(1 - 2^{-j})}{n}$$

where $\mathcal{C}_j$ is the number of clauses of length j. Now at a given node at depth i of a search tree, the search branches into a subtree of depth $n-i$ containing zero or more of the $\langle sol \rangle$ solutions. The constrainedness κ_i of the subproblems rooted at depth i may be re-estimated: the denominator of the fraction decreases by 1 at each level and the various clause counts $\mathcal{C}_j$ change as clauses are deleted or reduced in length. The principal observation in [3] is that as the search deepens, overconstrained problems appear to become *more* overconstrained, while underconstrained ones tend to become *less* constrained. Critically constrained problems seem to remain on a knife-edge of criticality until deep in the search tree. Figure 1 shows this phenomenon in an experiment with the Davis-Putnam algorithm DP (without pure literal deletion) for fixed clause length random 3-SAT problems.

A useful step towards understanding κ_i is to fix reasonably tight upper and lower bounds on it. A lower bound is given in [3] following the observation that, since the total number of solutions does not *increase* during the search, $\langle sol \rangle$ is an upper bound on the average number of solutions in the subtree into which a search branches at depth i. Therefore,

$$\kappa_i \geq 1 - \frac{\log_2 \langle sol \rangle}{n - i}$$

Now $\log_2 \langle sol \rangle = n(1 - \kappa)$, so

$$\kappa_i \geq \frac{n\kappa - i}{n - i}$$

To obtain an upper bound in a similar way, at least for methods that do not prune away solutions during the search, note that the maximum number of subproblems rooted at depth i is 2^i, so the average number of solutions per subproblem at that depth is at least $\langle sol \rangle / 2^i$. Therefore,

$$\kappa_i \leq 1 - \frac{\log_2 \langle sol \rangle - i}{n - i} = \frac{n\kappa}{n - i}$$

The lower bound exhibits the knife-edge very clearly: if the problem is underconstrained so that $\kappa < 1$ then the lower bound decreases as i increases; if it is overconstrained so that $\kappa > 1$ then the lower bound increases with i; if $\kappa = 1$ then the lower bound on κ_i is also 1 for all i. The upper bound, however, is less reassuring: as i increases, so does the upper bound on κ_i, regardless of the value of κ provided $\kappa > 0$. It therefore shows no knife-edge at all for positive values of κ.

It is suggested in [3] that this knife-edge phenomenon may help to explain why problems near the phase boundary tend to be hard. Underconstrained problems have many solutions, and the deeper the search goes the more plentiful they appear, so finding one becomes easy. Overconstrained problems have no solutions, and the deeper one searches the more evident it becomes that no solutions are present, so backtracking is quickly forced. On the knife-edge, however, it is not obvious whether there are solutions or not, and each time the search branches into a subproblem it looks similar to the problem before—neither more nor less constrained—until at last, deep in the search tree, the constrainedness breaks one way or the other and there is either a solution or a backtrack point.

4 Inside the constrainedness of search

As is familiar, a systematic backtracking search, such as that carried out by DP, traverses a tree. At the root, it has as axioms for a deductive theory the input clauses of the problem. At each node it reasons, checking the available clauses for consistency. The consistency check can only be partial, since it must carried out in low-order polynomial time. If no contradiction is detected, the remaining search space is partitioned: in the case of DP, by choosing a variable and adding it and its negation in turn as unit clauses. The additional clause makes more information available to the inference engine, so at the next level of the tree the partial consistency check is more likely to result in a contradiction. As soon as an inconsistency is detected, the search backtracks, terminating the branch. Intuitively, the process may be seen as a game in which at each point the current branch tries to grow by partitioning the search space and choosing one of the subspaces, while the constraints try to stop it by using the extra information provided by the partition to help detect an inconsistency. A solution is simply a branch which succeeds in growing to its full length n before the constraints manage to stop it.

A natural relaxation of this concept of a solution is that of a branch which succeeds in growing to a length k, for $k \leq n$. It is not obvious how the number of such "quasi-solutions" Q_k will vary with k, since at each level in the tree some subset of the nodes will be shown inconsistent and will die, while the others will each spawn two immediate successors.[3] Clearly the behaviour of Q_k depends on the problem and on the algorithm and any heuristics used. Equally clearly, however, $Q_0 = 1$ and, averaged over a set of problems over a state space of size 2^n, $Q_n = \langle sol \rangle$ regardless of the algorithm.

Usefully for the purpose of understanding κ_i, we may define a notion of local constrainedness, λ, similar to κ but with $\langle qsol \rangle_i$, the expected value of Q_i, in place of $\langle sol \rangle$:

$$\lambda_i = 1 - \frac{\log_2 \langle qsol \rangle_i}{i}$$

This represents the constrainedness *to* depth i, just as κ_i represents the constrainedness *from* depth i. The two are interestingly related to each other and to κ itself.

Since there are on average $\langle qsol \rangle_i$ subproblems rooted at depth i and $\langle sol \rangle$ solutions to be shared between them, the average subproblem at depth i has $\langle sol \rangle / \langle qsol \rangle_i$ solutions. Therefore, on average,

$$\kappa_i = 1 - \frac{\log_2 \langle sol \rangle - \log_2 \langle qsol \rangle_i}{n - i}$$

By the definitions of κ, and λ_i, that is to say

$$\kappa_i = 1 - \frac{n(1-\kappa) - i(1-\lambda_i)}{n-i} = \frac{n\kappa - i\lambda_i}{n-i}$$

or, where $p = i/n$ and $\bar{p} = 1 - p$,

$$\kappa = p\lambda_i + \bar{p}\kappa_i$$

That is, the constrainedness of a set of problems is the sum of their constrainedness *to* any depth and their constrainedness *from* that depth, weighted in proportion to the depth.

[3] For this purpose, unit propagation as applied in the Davis-Putnam algorithm is seen as resulting in extension not to a single successor node but to two, one of which dies immediately.

5 What does the knife-edge signify?

It is suggested above that the constrainedness knife-edge indicates criticality: that as the search goes deeper, κ_i moves further away from 1. That is, for a problem to remain on the knife-edge, it must remain critically constrained until near the bottom of the search tree, for as soon as it falls off the knife edge it will become more and more over or under constrained with increasing depth. It is also suggested that the same phenomenon contributes to the special hardness of critically constrained problems by forcing the search to go deep before decisions occur.

The lower bound on κ_i computed in §3 tends to support the association between the knife-edge and criticality, for its derivative with respect to i is

$$\frac{n(\kappa - 1)}{(n-i)^2}$$

which is positive, negative or zero according to whether κ is greater than, less than or equal to 1. However, differentiating the real κ_i with respect to i we get

$$\frac{n(\kappa - \lambda_i) - i(n-i)\lambda_i'}{(n-i)^2}$$

In other words, where p and $\bar{p}$ are as above,

$$\bar{p}\kappa_i' + p\lambda_i' = \frac{\kappa - \lambda_i}{n\bar{p}^2}$$

If the knife-edge is defined as the region in which constrainedness neither increases nor decreases with depth, then this is to be found where

$$\kappa - \lambda_i = np\bar{p}^2\lambda_i'$$

Since $\kappa - \lambda_i$ is rather small (always in the range 0 ± 1), for sufficiently large n this equation requires λ_i' to be close to zero, meaning that $\kappa - \lambda_i$ is almost constant, so that λ_i' is roughly on the order of $1/n$. As a special case, it holds wherever $\lambda_i = \kappa$ and $\lambda_i' = 0$. There are many other solutions of course. The important observation is that in general they do not require κ to be near 1.

Note that κ is algorithm-independent, but λ_i is not. κ_i might therefore reasonably be expected to depend on the algorithm too, whence we might well suspect that the shape and even the existence of the knife-edge will vary according to solution methods and heuristics. If that is so, then the knife-edge cannot be indicative simply of criticality.

In fact, it is so. Figure 2 shows an algorithm for solution of SAT problems. It follows DP up to the point of performing unit resolution and unit subsumption, but lacks unit propagation. This is very inefficient in comparison with the real Davis-Putnam algorithm, but nonetheless it is a legitimate SAT solver and behaves much like other complete search methods. In particular it exhibits a cost peak near the solvability phase transition in random k-SAT, just as DP does, and the shape of the peak is almost exactly the same in each case.

SDP does not, however, exhibit a knife-edge. Figure 3 shows the results of an experiment just like that reported in Figure 1 but using SDP in place of DP, and of course with smaller problems in order to get results in an acceptable time. It is a matter of taste whether we say that there is no knife-edge at all or that there is one at a clause to variable ratio of 0. Either way, there is none in the critically constrained region. It follows that the knife-edge phenomenon is by no means an inevitable product of the problem but depends on details of the search method used to solve it.

So the claim that the knife-edge indicates criticality remains unconvincing. What of the claim that it helps to explain the peculiar

```
function UNIT (S: clause-multiset,
               l: literal) : clause-multiset
  S_r ← S
  For each c ∈ S_r do
    If l ∈ c then S_r ← S_r \ [c]
    Else if l̄ ∈ c then c ← c \ {l̄}
  Return S_r
end of function UNIT
```

```
function SDP (S: clause-multiset,
              V: variable-set) : boolean
  If □ ∈ S then return FALSE
  If V = ∅ then return TRUE
  Randomly choose some p ∈ V
  If SAT (UNIT(S, p), V \ {p}) then return TRUE
  If SAT (UNIT(S, ¬p), V \ {p}) then return TRUE
  Return FALSE
end of function SAT
```

Figure 2. Simplified Davis-Putnam algorithm (SDP) for SAT

hardness of critically constrained problems? Again SDP casts doubt on that, for it shows a cost peak of the same shape as that for more efficient algorithms, yet it does so without a knife-edge. Is there some further aspect of phase transition behaviour in the case of algorithms such as DP for which the knife-edge furnishes at least a partial explanation?

To approach that question, first consider how the explanatory effect is supposed to work. As outlined above, the knife-edge of criticality is supposed to drive the search deep into its search tree before it finds backtrack points, thus increasing the tree depth. Since the width of the tree is exponential in its depth, that increases the total number of branches in the tree and also presumably the number in the portion of the tree which will actually be searched before a solution is found. An obvious experiment to check this account is to measure the size of search trees through the 3-SAT phase transition and check for any anomaly that may be attributed to, or have a common cause with, the knife-edge.

The DP algorithm solving random 3-SAT problems with 100 variables is again a suitable example. It is not feasible to traverse the search trees completely in very underconstrained cases, so we estimate the tree sizes by measuring the length of a sample of branches. Note that we are sampling branches from the *total* search tree (i.e. containing all solutions) in order to test the hypothesis that the knife-edge causes unexpectedly long branches near the phase transition; we are *not* directly measuring the search effort required to reach the first solution if any. The null hypothesis is that the tree size will decline as the constraints tighten, without any abrupt change near the phase boundary.

For the experiment, 500 problem instances were generated with c clauses for c ranging from 100 to 600 in steps of 5. 2000 branches were generated for each instance, using the standard MOMS heuristic to select variables for assignment but choosing randomly in each case whether to make the selected variable true or false. The size of the search tree for each instance was estimated as outlined in Appendix A, and the geometric mean of the 500 instances plotted. Figure 4 shows the results. For comparison, the curve has also been plotted for SDP without any heuristics. It is clear that both curves

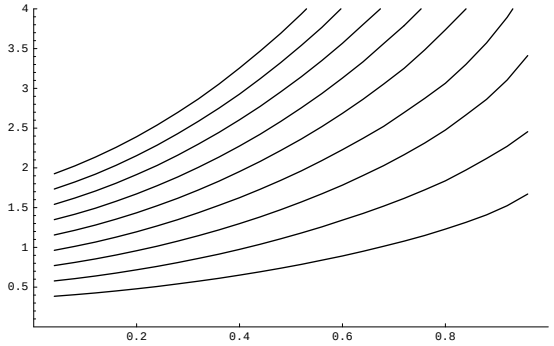

Figure 3. κ_i plotted against depth: SDP for 3-SAT, 25 variables.

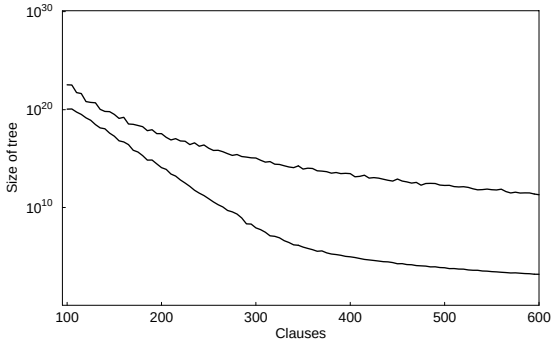

Figure 4. Geometric mean tree size for 3-SAT with 100 variables, plotted against number of clauses. Upper line: SDP; lower line: DP.

pass smoothly through the phase transition around 420–430 clauses with no hint of special behaviour at that point. Hence the knife-edge, where one is present, has no discernable effect on tree size (or the length of branches). We therefore conclude that the attractive thought that it helps to explain the cost peak, plausible as it may seem, is incorrect.

6 The knife-edge and search heuristics

Since the knife-edge is present in the case of the ordinary Davis-Putnam algorithm for SAT but absent when this is run without unit propagation, a natural thought is that the sharpness of the knife-edge may well correlate with the effectiveness of heuristics for a given problem class. This thought is given more weight by the consideration that the "perfect" knife-edge is that produced by the lower bound on κ_i enunciated in [3] and in §3 above, which is what would obtain if *per impossibile* all of the solutions were to be found on the only branch explored by the search.

Unfortunately for that natural thought, matters are not quite so simple. Figures 5 and 6 show the effect of enhancing DP with one of the best known refinements: pure literal deletion. This deletes any clause containing a literal which occurs only positively in the remaining problem. One result is that some solutions may be pruned away, along with the dross, for of course not all solutions need make the pure literal true. Pure literal deletion has little effect on overconstrained problems, but it brings huge efficiency gains on underconstrained ones. It is a highly effective addition to DP, yet figures 5 and 6 show that it tends somewhat to flatten out the knife-edge (on the underconstrained side) rather than to exaggerate it.

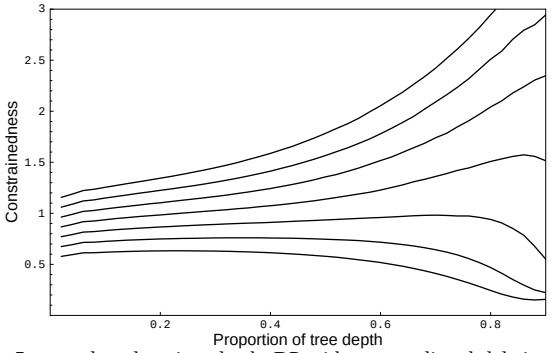

Figure 5. κ_i plotted against depth: DP without pure literal deletion for 3-SAT with 50 variables.

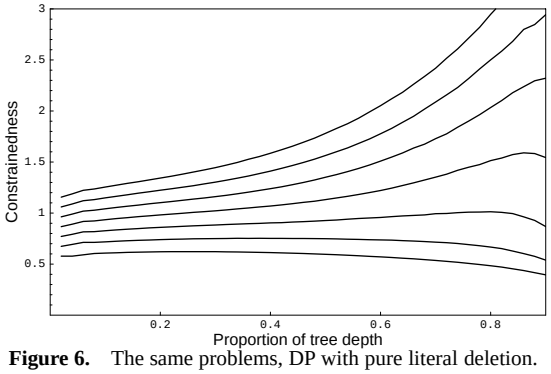

Figure 6. The same problems, DP with pure literal deletion.

7 Conclusions

The constrainedness knife-edge was an unexpected and striking discovery. The idea that the deeper the search goes the more clearly overconstrained and underconstrained problems reveal themselves as such is intuitively appealing, and the corollary that critically constrained problems are self-similar in respect of constrainedness hints at a new understanding of the relationship between criticality and difficulty.

The present paper shows that those thoughts, appealing as they may be, have to be rejected, or at least refined. The more closely we examine the phenomenon, the more puzzling it becomes: contrary to what at first looks obvious, it is not symptomatic of criticality, and experimental measurements of search trees show no evidence of a link between the knife-edge and hardness. At the same time, the regularity revealed in Figure 1 is not an illusion. The idea of splitting κ into λ_i and κ_i is simple, but the pattern observed in these latter points to something deep taking place *inside* search processes.

The obvious future direction for this research is to extend the experimental investigation, with a view to identifying the features which affect the presence, position and shape of a knife-edge. We are far from having a satisfactory theoretical explanation, though surely one is worth pursuing.

REFERENCES

[1] I. P. Gent, E. MacIntyre, P. Prosser, and T. Walsh, 'The constrainedness of search', in *Proc. AAAI-96*, pp. 246–252, (1996).
[2] E. Lozinskii, 'Information and evidence in logic systems', *Journal of Experimental and Theoretical Artificial Intelligence*, **6**, 163–193, (1994).
[3] T. Walsh, 'The constrainedness knife-edge', in *Proc. AAAI-98*, pp. 406–411, (1996).
[4] L. Wittgenstein, *Tractatus Logico-Philosophicus*, Routledge and Kegan Paul, 1922.

A Estimating tree size by sampling

The experiments reported in §6 call for the number of branches in a binary tree to be estimated from a sample. Sampling is easy: begin at the root, at each stage select a variable, randomly assign it true or false and propagate the constraints until a backtrack or a solution occurs. However, averaging the branch lengths to obtain the logarithm of the tree size gives a skewed result, short branches are overrepresented in the sample, since the probability of arriving at the end of a given branch of length i is 2^{-i}. The sum should therefore be weighted, each actual observation of a branch of length i counting as 2^i virtual observations.

Let obs_i be the number of actual observations of branches of length i, so $\text{obs}_i.2^i$ is the number of virtual observations of length i. Let OBS and VOBS be the totals of actual and virtual observations respectively. Let T be the number of branches in the tree, b_i the number which are of length i and p_i the proportion which are of length i, so trivially $b_i = p_i T$. Now in any binary tree,

$$\sum_i \frac{b_i}{2^i} = 1$$

which is to say

$$\sum_i \frac{p_i}{2^i} = T^{-1}$$

The sample gives us an estimate of p_i:

$$\widehat{p_i} = \frac{\text{obs}_i.2^i}{\text{VOBS}}$$

so, substituting in the previous equation:

$$\widehat{T^{-1}} = \sum_i \frac{\text{obs}_i}{\text{VOBS}}$$

or in other words

$$\widehat{T} = \frac{\text{VOBS}}{\text{OBS}}$$