Backtracking and Probing

Paul Walton Purdom, Jr., Indiana University

G. Neil Haven, Indiana University

Partial support provided by NSF Grant CCR 92-03942.

Abstract: We analyze two algorithms for solving constraint satisfaction problems. One of these algorithms, Probe Order Backtracking, has an average running time much faster than any previously analyzed algorithm for problems where solutions are common. Probe Order Backtracking uses a probing assignment (a preselected test assignment to unset variables) to help guide the search for a solution to a constraint satisfaction problem. If the problem is not satisfied when the unset variables are temporarily set to the probing assignment, the algorithm selects one of the relations that the probing assignment fails to satisfy and selects an unset variable from that relation. Then at each backtracking step it generates subproblems by setting the selected variable each possible way. It simplifies each subproblem, and tries the same technique on them. For random problems with v variables, t clauses, and probability p that a literal appears in a clause, the average time for Probe Order Backtracking is no more than v^n when $p \geq (\ln t)/v$ plus lower order terms. The best previous result was $p \geq \sqrt{(\ln t)/v}$. When the algorithm is combined with an algorithm of Franco that makes selective use of resolution, the average time for solving random problems is polynomial for all values of p when $t \leq O(n^{1/3}(v/\ln v)^{2/3})$. The best previous result was $t \leq O(n^{1/3}(v/\ln v)^{1/6})$. Probe Order Backtracking also runs in polynomial average time when $p \leq 1/v$, compared with the best previous result of $p \leq 1/(2v)$. With Probe Order Backtracking the range of p that leads to more than polynomial time is much smaller than that for previously analyzed algorithms.

1 Backtracking

The constraint satisfaction problem is to determine whether a set of constraints over discrete variables can be satisfied. Each constraint must have a form that is easy to evaluate, so any difficulty in solving such a problem comes from the interaction between the constraints and the need to find a setting for the variables that simultaneously satisfies all of the constraints.

Constraint satisfaction problems are extremely common. Indeed, the proof that a problem is NP-complete implies an efficient way to transform the problem into a constraint satisfaction problem. Most NP-complete problems are initially stated as constraint satisfaction problems. A few special forms of constraint satisfaction problems have known algorithms that solve problem instances in polynomial worst-case time. However, for the general constraint satisfaction problem no known algorithm is fast for the worst case.

When no polynomial-time algorithm is known for a particular form of constraint satisfaction problem, it is common practice to solve problem instances with a search algorithm. The basic idea of searching is to choose a variable and generate subproblems by assigning each possible value to the variable. In each subproblem the relations are simplified by plugging in the value of the selected variable. This step of generating simplified subproblems is called *splitting*. If any subproblem has a solution, then the original problem has a solution. Otherwise, the original problem has no solution. Subproblems that are simple enough (such as those with no unset variables) are solved directly. More complex subproblems are solved by applying the technique recursively.

If a problem contains the always *false* relation, then the problem has no solution. *Simple Backtracking* improves over plain search by immediately reporting no solution for such problems. Backtracking often saves a huge amount of time.

2 Probing

This paper considers two algorithms that are improvements over Simple Backtracking. Both algorithms use the idea of *probing*: if a fixed assignment to the unset variables solves the problem, no additional investigation is needed. Our algorithms probe by setting each unset variable to *false* and testing to see whether all relations simplify to *true*. The two probing algorithms are simple enough that it is possible to analyze their average running time.

Our first algorithm, *Backtracking with Probing*, uses backtracking, probing, and no additional techniques. In particular, during splitting it always picks the first unset variable from a fixed ordering on the variables.

Our second algorithm, *Probe Order Backtracking*, is more sophisticated in its variable selection. It has a fixed ordering on the variables and a fixed ordering on the relations. First, it checks that there are no always *false* relations. If an always *false* relation is encountered, the problem is not satisfiable and the algorithm backtracks. Next, it checks to see if there is a currently selected relation. If there is no currently selected relation, it selects the first relation that evaluates to *false* under the probing assignment. (If all clauses evaluate to *true* then the probing assignment solves the problem.) Finally, the algorithm does splitting using the first unset variable of the selected relation.

3 Probability Model

The average number of nodes generated when solving randomly generated problems is one measure of the quality of a search algorithm. We use this measure where our random problems are formed by the conjunction of independently generated random *clauses* (the logical *or* of literals, where a literal is a binary variable or the negation of a binary variable). A random clause is generated by independently selecting each literal with a fixed probability, p. We use v for the number of variables, and t for the number of clauses. For the asymptotic analysis, both p and t are functions of v.

Many algorithms have been analyzed with this random clause length model [3, 4, 7, 8, 12, 16, 18, 21, 22]. Most of these analyses and a few unpublished ones are summarized in [17]. A few algorithms have also been analyzed with the fixed length model, where random problems consist of random clauses of fixed length [1, 2, 14, 20]. This second probability model generates problems that are more difficult to satisfy but perhaps more like the problems encountered in practice. The second model leads to much more difficult analyses.

4 Summary of Results

This section summarizes the performance of Probe Order Backtracking and gives some intuition as to why Probe Order Backtracking is fast. The simpler Backtracking with Probing Algorithm turns out to provide no significant improvement over previously analyzed algorithms, so we do not discuss it in great detail.

This paper has contour plots showing the performance of probing algorithms. We also include contour plots for the approximate performance analyses of these algorithms. Each plot is for random problems with 50 variables. The vertical axis shows p, the probability that a given literal appears in a clause, running from 0.001 to 1 with ticks at 0.01 and 0.1. At $p = 0.01$ the average clause length for problems is 1. At $p = 0.1$ the average clause length is 10 literals. The horizontal axis shows t, the number of clauses, running from 2 to 250 with ticks at 10 and 100. When p is near 0 or 1 most problems are trivial. When p is low most problems are easy because they contain an empty clause; empty clauses are trivially unsatisfiable. When p is high most problems are easy because any assignment of values to variables is a solution to most problems. The region of hard problems lies in the middle.

In most cases the contours are shaped like elongated horseshoes (see Fig. 2). The area within a horseshoe contour represents problems that are more difficult than the problems outside the contour. The outermost contour shows where the average number of nodes is 50, the next inner one 50^2, next 50^3, and finally 50^4. Running near the centerline of the horseshoes is a line that shows for each t that value of p that results in the hardest problems (those with the largest number of nodes).

In the less favorable cases the upper and lower branches of a contour do not meet (see Fig. 1). In those cases the uppermost and lowermost lines show where there is an average of 50 nodes, the next inner pair 50^2 nodes, and so on. Again the centerline shows the p value that results in the largest number of nodes. Occasionally one of the contours runs along one of boundaries (see Fig. 6). The contours do not always extend to the right edge of the figure due to difficulties with floating point overflow (see Fig. 1).

Figure 1 is a contour plot of the average number of nodes generated by Backtracking with Probing. This plot show shows that Backtracking with Probing provides no significant improvement over previously analyzed algorithms [17].

Figure 2 is a contour plot of the average number of nodes generated by Probe Order Backtracking. In this case the upper and lower contours join to form horseshoe shaped curves. Note that the region of hard problems is considerably smaller for Probe Order Backtracking than for Backtracking with Probing. Except

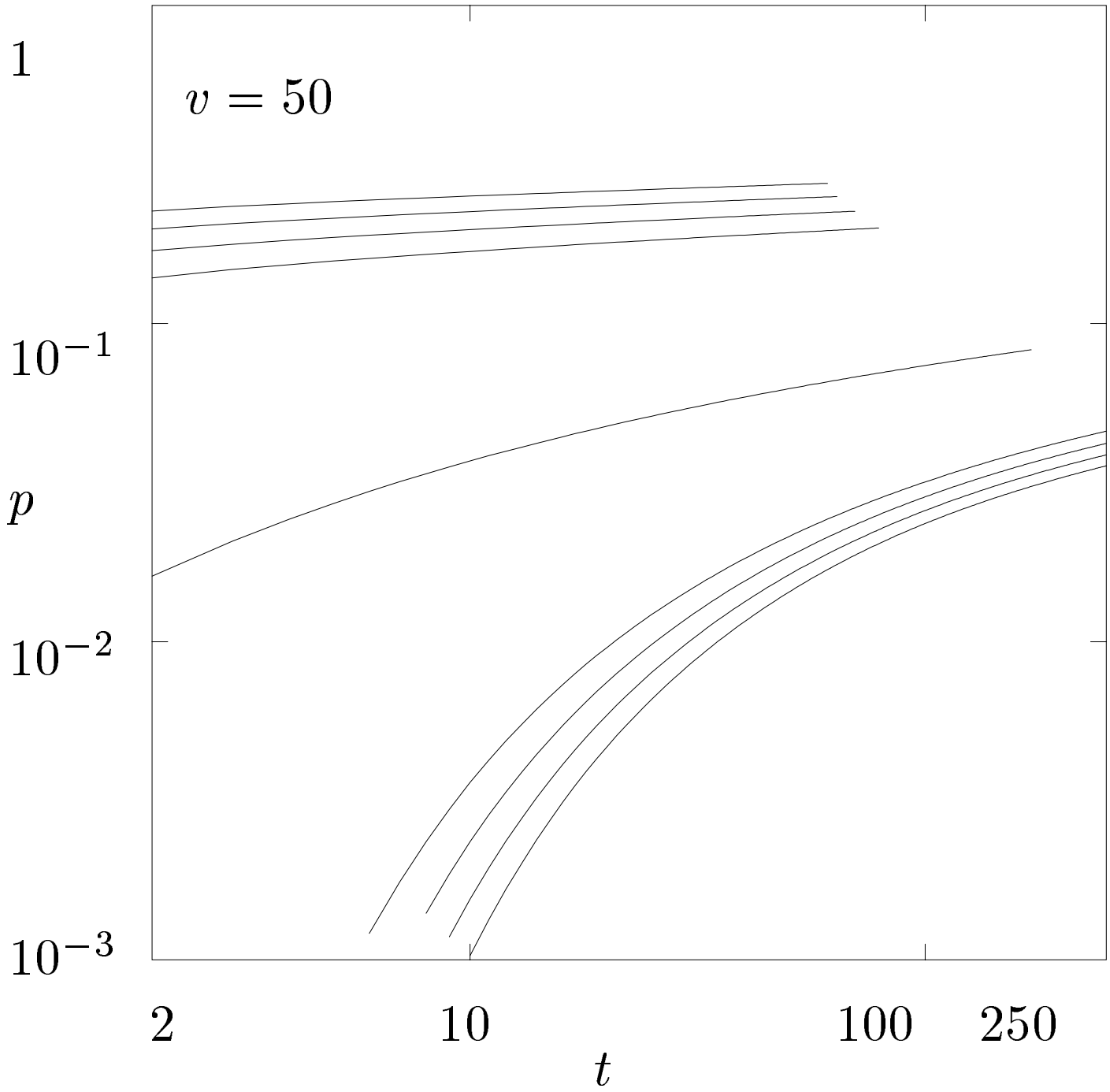

Fig. 1. Backtracking with Probing.

for the small t region, these contours are much better than those of any other algorithm for which such contours have been published. The improvement is particularly noticeable along the upper contour.

Figure 3 shows how the average number of nodes for Probe Order Backtracking compares with the average for several other satisfiability algorithms when, for each value of t and each algorithm, p is set to make the average as large as possible. The horizontal axis is the number of clauses (from 1 to 500 for this graph). The vertical axis is the average number of nodes from 1 to over 10^{15} with tick marks for each power of 10. Of the curves that were computed to $t = 500$, the uppermost is Goldberg's simplified version of the pure literal rule [9, 10], next is Clause Order Backtracking [3], and lowermost is backtracking combined with Goldberg's version of the pure literal rule [19]. Of the curves that stop short of $t = 500$ the highest for large

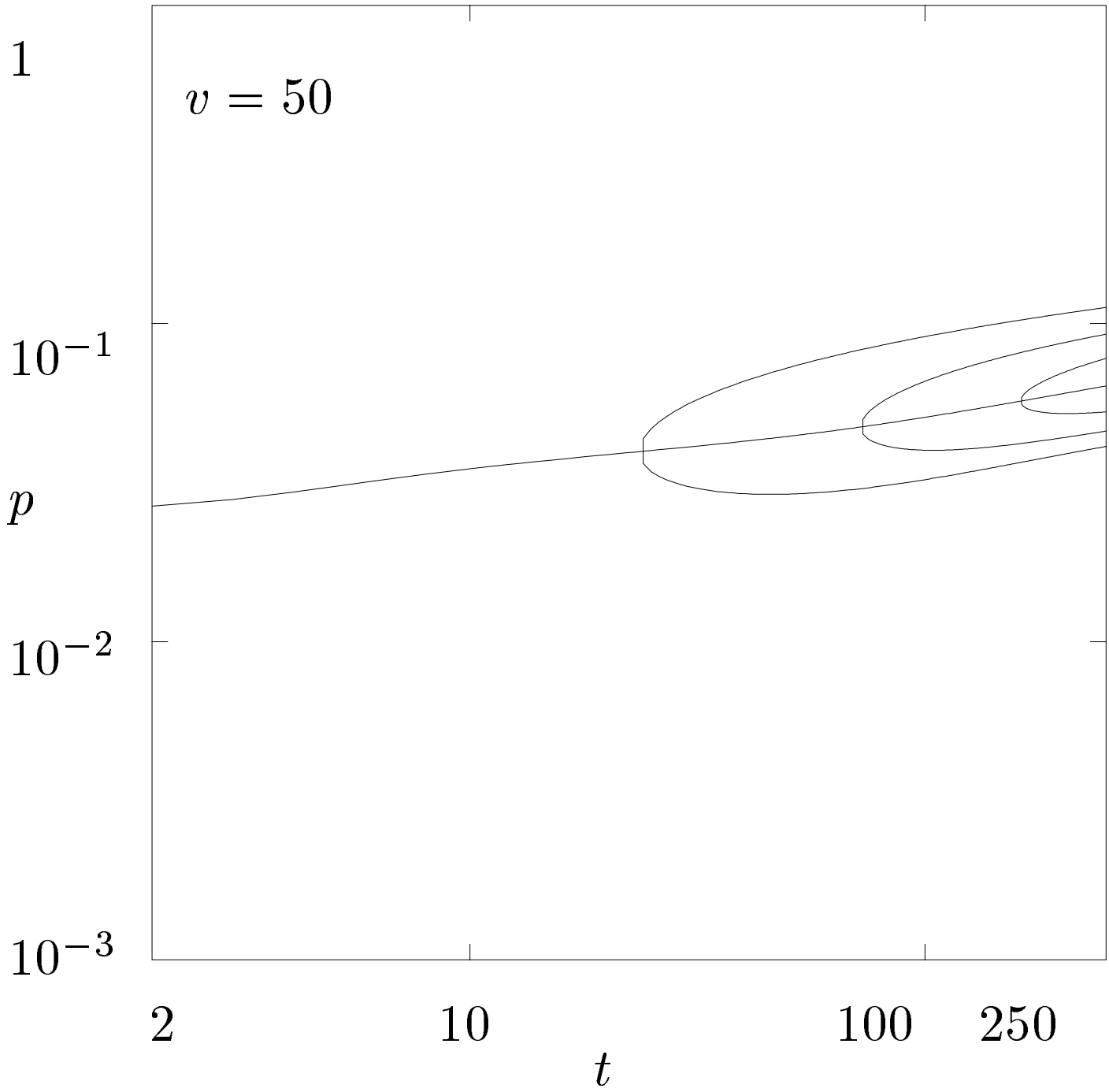

Fig. 2. Probe Order Backtracking.

t is the Full Pure Literal Rule [18], next is the Full Pure Literal Rule modified to ignore tautological clauses [18], next is Probe Order Backtracking [this paper], and lowest is Probe Order Backtracking combined with Goldberg's version of the Pure Literal Rule [11].

These curves show that there are huge differences in the average number of nodes generated by the various satisfiability algorithms. The average time for the Probe Order Backtracking-type algorithms is by far the best among the analyzed algorithms when t/v is not small.

The asymptotic analysis of Probe Order Backtracking shows that the average number of nodes is no more than v^n, for large v and $n > 1$, when any of the following conditions hold

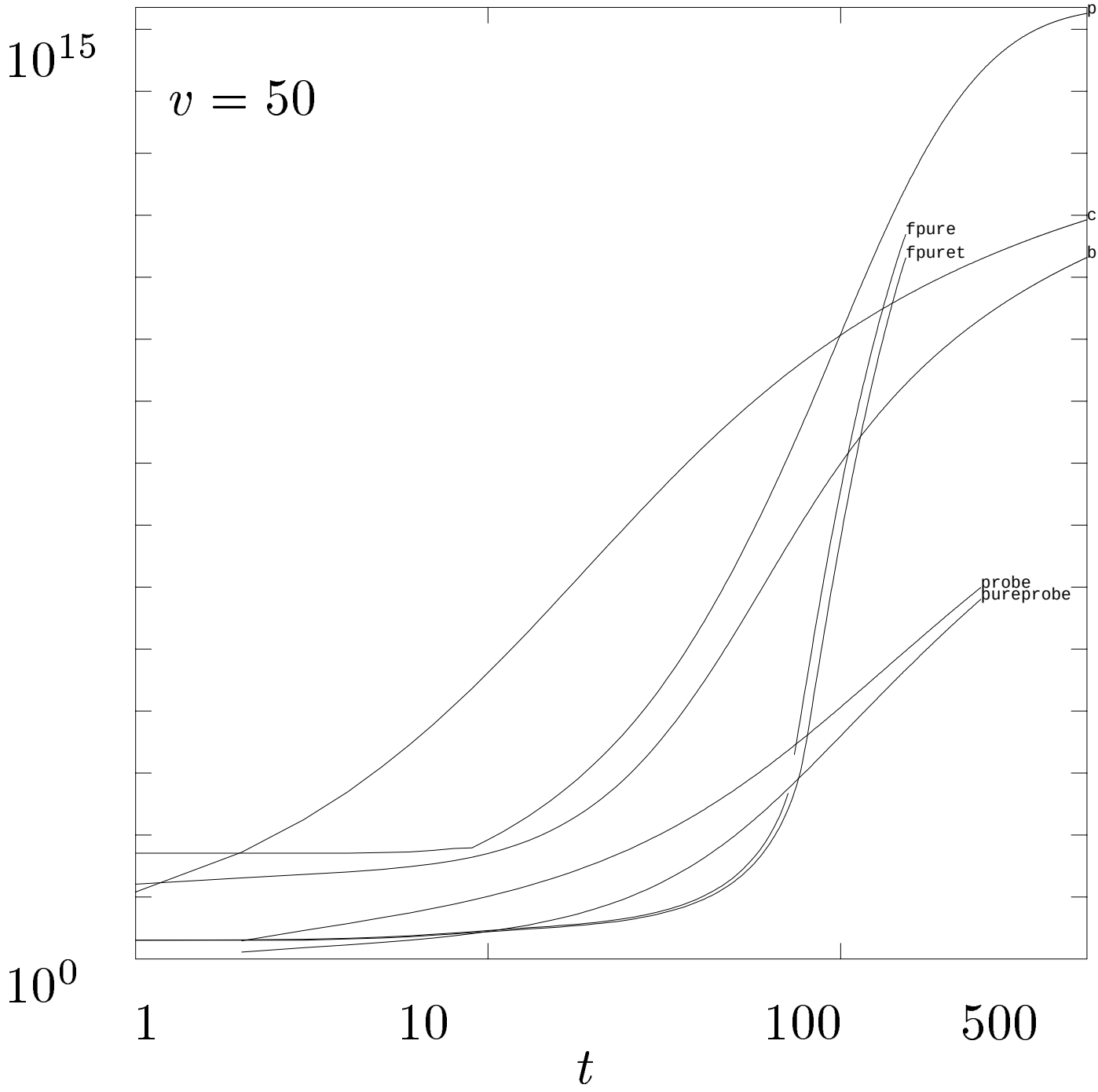

Fig. 3. Worst p performance.

$$p \geq \frac{\ln t + 2\ln\ln t - \ln\ln v - \ln(n-1) - \ln 2}{v} + \Theta\left(\frac{\ln\ln t}{v\ln t}\right), \tag{1}$$

$$\begin{aligned} p \leq & \left\lceil \frac{\ln(t/v) + \ln\ln(t/v) + 1 - \ln 2 - \ln\ln 2}{2v} \right\rceil \\ & \times \left[1 + \frac{2(n-1)\ln v - (\ln 2)[\ln(t/v) + \ln\ln(t/v) + 3 - \ln 2 - \ln\ln 2]}{4v\ln 2} \right. \\ & \left. - \Theta\left(\frac{(\ln v)\ln(t/v)}{v^2}\right) - \Theta\left(\frac{\ln\ln(t/v)}{v\ln(t/v)}\right)\right], \end{aligned} \tag{2}$$

$$p \le \frac{1}{v} + \frac{e[(n-1)\ln v - \ln 2]}{tv}, \tag{3}$$

$$t \le \frac{(n-1)\ln v - \ln 2}{\ln\{1 + [(pv)^2 - 1]e^{-pv}/2\}}. \tag{4}$$

The Θ term in bound (1) requires that t increases more rapidly than $\ln v$. Bound (4) requires the assumption that the limit of pv is greater than 1 and finite.

By setting p to minimize the right side of (4) we see that for all p and large v, the average number of nodes is no more than v^n when

$$t \le 5.1150(n-1)\ln v - \Theta(1). \tag{5}$$

Details: This sentence shows that you are reading the technical report. The extra details of the technical report are contained in sections starting with the word 'details' and ending with a diamond.$\diamond$

Details: In bound (4) the minimum value of the right side (when considered as a function of pv) occurs at $pv = 1 + \sqrt{2}$.$\diamond$

The bound for large p, bound (1), is much better than that for any previously analyzed algorithm. The best previous result was

$$p \ge \sqrt{\frac{\ln t - \ln n}{v}} \tag{6}$$

for Iwama's inclusion-exclusion algorithm [12]. In the region between bounds (1) and (6) Probe Order Backtracking is the fastest algorithm with proven results on its running time. Algorithms that repeatedly adjust variable settings to satisfy as many clauses as possible [24] are even faster on many problems. Those algorithms, however, have difficulty with problems which have no solution. They have been difficult to correctly analyze, and it is not clear at this time what their average running time is.

For the best previously analyzed algorithms there was a large range of p where the algorithms apparently required more than polynomial time. (The word apparently is used because the analyses were all upper bound analyses.) The ratio of the large p boundary to the small p boundary was $v^{1/2}$ times logarithmic factors. For Probe Order Backtracking, only the logarithmic factors are left. In some cases even the logarithmic factors are gone and the ratio is constant (in the limit of large v). Bound (2) for small p results from the fact that the average number of nodes for Probe Order Backtracking is no larger than the average for Simple Backtracking. When $t = v^\alpha$ with $\alpha > 1$, the ratio of the upper boundary (1) to the lower boundary (2) is $2\alpha/(\alpha - 1)$ plus lower order terms. Thus, for large α only a very limited range of p leads to problems with a large average time.

Perhaps the region of greatest interest is the one where t is proportional to v. When t is below $3.22135v$, bound (3) is better than (2). When $t = \beta v$ the ratio of the upper bound (1) to first lower bound (2) is $(2\ln v)/(\ln\beta + \ln\ln\beta + 1 - \ln 2 - \ln\ln 2)$ plus lower order terms. The ratio of the upper bound to the second lower bound (3) is $\ln v$ plus lower order terms.

Previously, for small t the best algorithm was a combination of Franco's limited resolution algorithm [8] for small p and Iwama's inclusion-exclusion algorithm [12] for large p. When p is unknown, the two algorithms can be run in parallel and stopped as soon as an answer is found. Each algorithm generates no more than than v^n nodes (regardless of p) when $t = O(n^{1/3}(v/\ln v)^{1/6})$. Combining Franco's algorithm with Probe Order Backtracking improves the bound to $O(n^{1/3}(v/\ln v)^{2/3})$. The techniques of Franco's algorithm can be combined with Probe Order Backtracking, so there is no longer any need to have two algorithms running in parallel. This is helpful when designing practical algorithms.

The basic idea behind probing is old. The idea resembles that used by Newell and Simon in GPS [15]. Just as their program concentrates on differences between its current state and its goal state, Probe Order Backtracking focuses on a set of troublesome relations that are standing in the way of finding a solution. It appears that people who are good at solving puzzles use related ideas all the time.

Franco observed that two extremely simple algorithms could quickly solve most problems outside of a small range of p [6]. His algorithm for the region of high p did a single probe and gave up if no solution was found. His algorithm for the region of low p looked for an empty clause and gave up if there was none. Since Franco's algorithms sometimes gave up, their average time was not well defined.

At the time of Franco's work it was already known that Simple Backtracking was fast along the lower boundary (2), but it was not clear how to obtain an algorithm with a fast average time along the upper

boundary (1). Simple uses of probing did not seem to lead to a good average time. Probe Order Backtracking was discovered while considering Franco's results [6] and considering the measurements of Sosič and Gu [24] for algorithms that concentrate on adjusting values until a solution is found. Both of those algorithms have difficulty with problems that have no solution.

Simple Backtracking improves over plain search by noticing when a problem has no solution due to the presence of an empty clause. However, Simple Backtracking is unfocused in its variable selection. So long as a problem does not have an empty clause, Simple Backtracking always proceeds by selecting the next splitting variable from a fixed ordering. The Clause Order Backtracking Algorithm [3] improves over Simple Backtracking by focusing on the variables in one clause of the problem at a time. This method of searching has the advantage that it performs splitting on just those variables that actually appear in a problem.

The Clause Order Backtracking Algorithm provides a framework for the construction of a probing algorithm that has good performance for a wide range of problems, including those with no solution. Probe Order Backtracking, like Clause Order Backtracking, focuses on the variables in one clause at a time. However, Probe Order Backtracking improves over Clause Order Backtracking by only selecting variables from clauses which are not satisfied by the probing assignment. These are the clauses standing in the way of finding a solution. Our simpler algorithm, Backtracking with Probing, lacks this feature; like the Simple Backtracking Algorithm it selects variables from a fixed ordering. Its only use of probing is to test for a solution before picking a new variable for splitting. The analysis of Backtracking with Probing shows that such a naive application of probing does not lead to fast average time for the region of high p or for the region of low t. For good performance it appears to be essential that an algorithm use probing both to notice when there is a solution and to indicate which clauses are interfering with solving the problem.

The focused nature of Probe Order Backtracking's search often leads to a rapid solution of a problem. Of course, setting variables to satisfy one relation sometimes causes other relations to become unsatisfied. In the worst case, the algorithm may need to try almost every combination of values for the variables. Thus, the average-case performance of Probe Order Backtracking is extremely good, but its worst-case performance is not an improvement over previous algorithms.

5 Practical Algorithms

Probe Order Backtracking was studied in part because it is simple enough to analyze. In practice one wants an algorithm that is fast whether or not it is possible to analyze its running time. There are several improvements that would clearly improve Probe Order Backtracking's average speed even though it is difficult to analyze their precise effectiveness:

1. Stop the search as soon as one solution is found. The analysis suggests that this would greatly improve the speed near the upper boundary (1), but stopping at the first solution leads to statistical dependencies that are difficult to analyze.
2. Carefully choose the probing sequence instead of just setting all variables to a fixed value. Various greedy approaches where variables are set to satisfy as many clauses as possible should be considered (see [13, 24]). This is particularly important near the upper boundary (1).
3. Probe with several sequences at one time. See [5, p 151] for an algorithm that used two sequences. This is helpful along the upper boundary.
4. Carefully select which variable to set. The analysis suggests that this is particularly important along the lower boundary. Variables in hard to satisfy relations (short clauses) are more important than those in easy to satisfy relations. Variables that appear in lots of relations are more important than those that appear in a few relations. Apparently when the relations are clauses it is helpful to consider the number of clauses containing a particular variable positively and the number containing it negatively [5]. It appears that variable selection was a major factor in determining the order of placement of winning entries in a recent SAT competition [5].
5. Use resolution when it does not increase the problem size [8].

6 Algorithm Statement

The precise form of Probe Order Backtracking that is analyzed along with the rules for charging time is given below. This version of the algorithm is specialized to work on satisfiability problems presented in conjunctive normal form. The Backtracking with Probing Algorithm is a modification of this algorithm.

A literal is *positive* if it is not in the scope of a *not* sign. It is *negative* if it is in the scope of a *not* sign. In the following algorithm a variable can have the value *true*, *false*, or *unset*. The *positively-augmented* current assignment is the current assignment of values to variables with the *unset* values changed to *true*. The *negatively-augmented* current assignment is obtained by setting the *unset* values to *false*.

The algorithm simplifies clauses by plugging in the values of the set variables, so that (except when simplifying) it is concerned only with those variables that have the value *unset*. In this algorithm the set of solutions is a global variable that is initially the empty set. Any solutions that are found are added to the set. If the problem has any solutions, at least one solution will be added to the set before the algorithm terminates. The algorithm may find more than one solution, but it does not in general find all solutions. If the problem has no solution, then the algorithm will terminate with an empty set of solutions. Notice that the algorithm ignores tautological clauses.

Probe Order Backtracking for CNF problems.

1. (Empty.) If the CNF problem has an empty clause, then return (no solution), and charge one time unit.
2. (Probe.) If there are no all positive clauses (that is, every clause has at least one negative literal), then return with the negatively-augmented current assignment added to the set of solutions and charge one time unit.
3. (Trivial.) If every clause of the CNF problem has only positive literals then return with the positively-augmented current assignment added to the set of solutions and charge one unit of time.
4. (Select.) Choose the first clause that is all positive. Step 2 ensures that there is at least one such clause.
5. (Splitting.) Let k be the number of variables in the selected clause. (Step 1 ensures that $k \geq 1$, and Step 4 ensures that each variable occurs in at most one literal of the clause.) For j starting at 1 and increasing to at most k, generate the j^{th} subproblem by setting the first $j-1$ variables of the clause so that their literals are *false* and setting the j^{th} variable so that its literal is *true*. Use the assignment of values to simplify the problem (remove each false literal from its clause and remove from the problem each clause with a true literal). Apply the algorithm recursively to solve the simplified problem. If setting the first $j-1$ literals of the selected clause to *false* results in some clause being empty, then stop generating subproblems. If the loop stops with $j = h$, then charge $h+1$ time units.

The cost in time units has been defined to be the same as the number of nodes in the backtrack tree generated by the algorithm. The actual running time of the algorithm depends on how cleverly it is implemented, but a good implementation will result in a time that is proportional to the number of nodes multiplied by a factor that is between 1 and tv, where v is the number of variables and t is the number of clauses.

The backtrack tree includes nodes for determining that the selected clause is empty. The computation associated with those nodes can be done quickly, so one might wish to have an upper limit of k on the time units for Step 5. This would lead to small, unimportant changes in the analysis.

Backtracking with Probing replaces Steps 4 and 5 with a step that selects the first unset variable and generates two subproblems: one where the selected variable is set to *false* and one where it is set to *true*.

7 Exact Analysis

The remainder of this paper consists of the analyses of the Backtracking with Probing Algorithm and the Probe Order Backtracking Algorithm. Since the Backtracking with Probing Algorithm does not offer any significant improvement over other previously analyzed algorithms, we restrict our asymptotic analysis to the Probe Order Backtracking Algorithm. The reader who wants more detailed analyses should refer to [23].

We now derive recurrence equations which give exact values of the average number of nodes generated by each algorithm.

7.1 Basic Probabilities

For analysis of probing algorithms it is useful to divide clauses into the following categories: *empty* (no literals), *all positive* (1 or more positive literals), *tautological* (a positive and negative literal for the same variable, possibly with additional literals), and *other* (any clause that does not fall into one of the preceding

categories). Assigning values to some variables and then simplifying the clause may change the category of a clause, or it may result in the clause becoming *satisfied*. (Note that empty clauses remain empty and all positive clauses never become other clauses.)

The probability that a random clause formed from v variables is nontautological, contains j positive literals, and contains k negative literals is

$$P(v,j,k) = \binom{v}{j,k,v-j-k} p^{j+k}(1-p)^{2v-j-k}. \tag{7}$$

The probability that a random clause has no literals is

$$P(v,0,0) = (1-p)^{2v}. \tag{8}$$

Note that

$$\sum_{j,k} P(v,j,k) = (1-p^2)^v \leq 1, \tag{9}$$

because tautological clauses are not counted in the double sum.

Details:

$$\sum_j P(v,j,k) = \binom{v}{k} p^k (1-p)^v \sum_j \binom{v-k}{j} p^j (1-p)^{v-k-j} = \binom{v}{k} p^k (1-p)^v.$$

$$\sum_{j,k} P(v,j,k) = \sum_k \binom{v}{k} p^k (1-p)^v = (1-p)^v (1+p)^v = (1-p^2)^v.$$

◇

Suppose you form a random clause from v variables and then select one of the v variables at random. The probability that the clause has a particular value of j and k (implying that it is not a tautology) and that the selected variable appears in the indicated way is

$$\text{positive} : \frac{j}{v} P(v,j,k), \quad \text{negative} : \frac{k}{v} P(v,j,k), \quad \text{neither} : \frac{v-j-k}{v} P(v,j,k). \tag{10}$$

7.1.1 All-Positive Clauses

The probability that a random clause is all positive is

$$\sum_{j \geq 1} P(v,j,0) = (1-p)^v [1-(1-p)^v]. \tag{11}$$

Details:

$$\sum_{j \geq 1} P(v,j,0) = \sum_{j \geq 0} P(v,j,0) - P(v,0,0) = (1-p)^v - (1-p)^{2v}.$$

◇

Suppose clauses are generated at random until an all positive clause is produced. The probability that the all positive clause contains j literals is

$$A(v,j) = \frac{P(v,j,0)}{\sum_{j \geq 1} P(v,j,0)} = \binom{v}{j} \frac{p^j (1-p)^{v-j}}{1-(1-p)^v}. \tag{12}$$

If a random variable is assigned the value *true*, then an all positive clause will either become satisfied or remain all positive. The probability that the clause will become satisfied is

$$\sum_j \frac{j}{v} A(v,j) = \frac{p}{1-(1-p)^v}. \tag{13}$$

Details:

$$\begin{aligned}\sum_{j\geq 1}\frac{j}{v}A(v,j) &= \sum_{j\geq 0}\frac{j}{v}\binom{v}{j}\frac{p^j(1-p)^{v-j}}{1-(1-p)^v}\\ &= \frac{p}{1-(1-p)^v}\sum_{j\geq 0}\binom{v-1}{j-1}p^{j-1}(1-p)^{v-1-(j-1)}\\ &= \frac{p}{1-(1-p)^v}.\end{aligned}$$

◇

The probability that the clause has length j and that it remains all positive is

$$\frac{v-j}{v}A(v,j) = \binom{v-1}{j}\frac{p^j(1-p)^{v-j}}{1-(1-p)^v} = \frac{P(v-1,j,0)}{(1-p)^{v-2}[1-(1-p)^v]}. \tag{14}$$

If a random variable is assigned the value *false*, then an all positive clause will either become empty or remain all positive. The probability that the resulting clause will be empty is

$$\frac{1}{v}A(v,1) = \frac{p(1-p)^{v-1}}{1-(1-p)^v}. \tag{15}$$

The probability that the resulting clause will be all positive with length $j \geq 1$ is

$$\frac{j+1}{v}A(v,j+1) + \frac{v-j}{v}A(v,j) = \frac{P(v-1,j,0)}{(1-p)^{v-1}[1-(1-p)^v]}. \tag{16}$$

Details:

$$\begin{aligned}\frac{j+1}{v}A(v,j+1) + \frac{v-j}{v}A(v,j) &= \frac{j+1}{v}\binom{v}{j+1}\frac{p^{j+1}(1-p)^{v-j-1}}{1-(1-p)^v} + \frac{v-j}{v}\binom{v}{j}\frac{p^j(1-p)^{v-j}}{1-(1-p)^v}\\ &= \binom{v-1}{j}\frac{p^j(1-p)^{v-j-1}}{1-(1-p)^v}[p+1-p]\\ &= \binom{v-1}{j}\frac{p^j(1-p)^{v-j-1}}{1-(1-p)^v} = \frac{P(v-1,j,0)}{(1-p)^{v-1}[1-(1-p)^v]}.\end{aligned}$$

◇

The average length of a random all-positive clause is

$$\sum_j jA(v,j) = \frac{pv}{1-(1-p)^v}. \tag{17}$$

Details: Multiply eq. (13) by v.◇

7.1.2 Other Clauses

The probability that a random clause is an other clause is

$$\sum_{\substack{j\geq 0\\ k\geq 1}} P(v,j,k) = (1-p)^v[(1+p)^v - 1]. \tag{18}$$

Details:

$$\sum_{k\geq 1}\sum_j P(v,j,k) = \sum_k \binom{v}{k}p^k(1-p)^v - (1-p)^v = (1-p)^v[(1+p)^v - 1].$$

◇

Suppose clauses are generated at random until an other clause is produced. The probability that the other clause contains j positive literals and $k \geq 1$ negative literals is

$$M(v,j,k) = \frac{P(v,j,k)}{\sum_j \sum_{k\geq 1} P(v,j,k)} = \binom{v}{j,k,v-j-k} \frac{p^{j+k}(1-p)^{v-j-k}}{(1+p)^v - 1}. \tag{19}$$

If a random variable is assigned the value *true*, an other clause may become empty, become all positive, become satisfied, or remain an other clause. The probability that an other clause will become an empty clause is

$$\frac{1}{v} M(v,0,1) = \frac{p(1-p)^{v-1}}{(1+p)^v - 1}. \tag{20}$$

The probability that it will become an all positive clause with length j is

$$\frac{1}{v} M(v,j,1) = \binom{v-1}{j} \frac{p^{j+1}(1-p)^{v-j-1}}{(1+p)^v - 1} = \frac{pP(v-1,j,0)}{(1-p)^{v-1}[(1+p)^v - 1]}. \tag{21}$$

The probability that it will be satisfied is

$$\sum_j \sum_{k\geq 1} \frac{j}{v} M(v,j,k) = \frac{p[(1+p)^{v-1} - 1]}{(1+p)^v - 1} \tag{22}.$$

Details:

$$\begin{aligned}
\sum_j \sum_{k\geq 1} \frac{j}{v} M(v,j,k) &= \sum_j \sum_{k\geq 1} \frac{j}{v} \frac{v!}{j!k!(v-j-k)!} \frac{p^{j+k}(1-p)^{v-j-k}}{(1+p)^v - 1} \\
&= \sum_j \sum_{k\geq 1} \frac{(v-1)!}{(j-1)!k!(v-j-k)!} \frac{p^{j+k}(1-p)^{v-j-k}}{(1+p)^v - 1} \\
&= \sum_{k\geq 1} \binom{v-1}{k} p^k \sum_j \binom{v-1-k}{j-1} \frac{p^j (1-p)^{v-j-k}}{(1+p)^v - 1} \\
&= \sum_{k\geq 1} \binom{v-1}{k} \frac{p^{k+1}}{(1+p)^v - 1} = \frac{p[(1+p)^{v-1} - 1]}{(1+p)^v - 1}.
\end{aligned}$$

◇

The probability that it will become an other clause with j positive literals and $k \geq 1$ negative literals is

$$\frac{k+1}{v} M(v,j,k+1) + \frac{v-j-k}{v} M(v,j,k) = \frac{P(v-1,j,k)}{(1-p)^{v-1}[(1+p)^v - 1]}. \tag{23}$$

Details:

$$\begin{aligned}
&\frac{k+1}{v} M(v,j,k+1) + \frac{v-j-k}{v} M(v,j,k) \\
&\qquad = \frac{k+1}{v} \frac{v!}{j!(k+1)!(v-j-k-1)!} \frac{p^{j+k+1}(1-p)^{v-j-k-1}}{(1+p)^v - 1} \\
&\qquad\quad + \frac{v-j-k}{v} \frac{v!}{j!k!(v-j-k)!} \frac{p^{j+k}(1-p)^{v-j-k}}{(1+p)^v - 1} \\
&\qquad = \frac{(v-1)!}{j!k!(v-j-k-1)!} \frac{p^{j+k}(1-p)^{v-j-k-1}}{(1+p)^v - 1} [p + 1 - p] \\
&\qquad = \frac{(v-1)!}{j!k!(v-j-k-1)!} \frac{p^{j+k}(1-p)^{v-j-k-1}}{(1+p)^v - 1} = \frac{P(v-1,j,k)}{(1-p)^{v-1}[(1+p)^v - 1]}.
\end{aligned}$$

◇

If a random variable is assigned the value *false*, an other clause may become become satisfied or remain an other clause. The probability that an other clause will be satisfied is

$$\sum_j \sum_{k \geq 1} \frac{k}{v} M(v,j,k) = \frac{p(1+p)^{v-1}}{(1+p)^v - 1}. \tag{24}$$

Details:

$$\begin{aligned}
\sum_j \sum_{k \geq 1} \frac{k}{v} M(v,j,k) &= \sum_j \sum_{k \geq 1} \frac{k}{v} \frac{v!}{j!k!(v-j-k)!} \frac{p^{j+k}(1-p)^{v-j-k}}{(1+p)^v - 1} \\
&= \sum_j \sum_{k \geq 1} \frac{(v-1)!}{j!(k-1)!(v-j-k)!} \frac{p^{j+k}(1-p)^{v-j-k}}{(1+p)^v - 1} \\
&= \sum_{k \geq 1} \binom{v-1}{k-1} p^k \sum_j \binom{v-k}{j} \frac{p^j (1-p)^{v-j-k}}{(1+p)^v - 1} \\
&= \sum_{k \geq 1} \binom{v-1}{k-1} \frac{p^k}{(1+p)^v - 1} = \frac{p[(1+p)^{v-1} - 1]}{(1+p)^v - 1}.
\end{aligned}$$

◇

The probability that it will become an other clause with j positive literals and $k \geq 1$ negative literals is

$$\frac{j+1}{v} M(v,j+1,k) + \frac{v-j-k}{v} M(v,j,k) = \frac{P(v-1,j,k)}{(1-p)^{v-1}[(1+p)^v - 1]}. \tag{25}$$

Details:

$$\begin{aligned}
&\frac{j+1}{v} M(v,j+1,k) + \frac{v-j-k}{v} M(v,j,k) \\
&\quad = \frac{j+1}{v} \frac{v!}{(j+1)!k!(v-j-k-1)!} \frac{p^{j+k+1}(1-p)^{v-j-k-1}}{(1+p)^v - 1} \\
&\qquad + \frac{v-j-k}{v} \frac{v!}{j!k!(v-j-k)!} \frac{p^{j+k}(1-p)^{v-j-k}}{(1+p)^v - 1} \\
&\quad = \frac{(v-1)!}{j!k!(v-j-k-1)!} \frac{p^{j+k}(1-p)^{v-j-k-1}}{(1+p)^v - 1} [p + 1 - p] \\
&\quad = \frac{(v-1)!}{j!k!(v-j-k-1)!} \frac{p^{j+k}(1-p)^{v-j-k-1}}{(1+p)^v - 1} = \frac{P(v-1,j,k)}{(1-p)^{v-1}[(1+p)^v - 1]}.
\end{aligned}$$

◇

Eqs. (14, 16, 21, 23, and 25) show that in all cases where a nonempty clause results from setting a variable associated with a random nonempty clause generated from v variables, the resulting clause has the same relative distribution as random clauses generated from $v-1$ variables. Thus, it is possible to base an analysis on the number of all positive clauses, the number of other clauses, and the number of variables without having to contend with statistical dependencies.

7.1.3 Total Number of Nodes

Eq. (8) implies that a random predicate with t clauses contains an empty clause (and is therefore solved with one node) with probability

$$1 - [1 - (1-p)^{2v}]^t. \tag{26}$$

Eqs. (8, 11, 18) imply that the probability that a random predicate contains zero empty clauses, m all positive clauses, n other clauses, and $t-m-n$ tautological clauses is

$$\binom{t}{m, n, t-m-n} (1-p)^{(n+m)v} [1 - (1-p)^v]^m [(1+p)^v - 1]^n [1 - (1-p)^v (1+p)^v]^{t-m-n}. \tag{27}$$

Details: The probability that a clause is empty is $(1-p)^{2v}$.
The probability that a clause is all positive is $(1-p)^v[1-(1-p)^v]$.
The probability that a clause is an other clause is $(1-p)^v[(1+p)^v-1]$.
The probability that a clause is none of the above, and hence tautological, is

$$1-(1-p)^{2v}-(1-p)^v[1-(1-p)^v]-(1-p)^v[(1+p)^v-1]=1-(1-p)^v(1+p)^v.$$

$\diamond$

If we let $T(v,m,n)$ be the average time required to solve a random problem with v variables, m all positive clauses, n other clauses, and no empty clauses then by summing all of the cases we see that the expected number of nodes is

$$1-[1-(1-p)^{2v}]^t+\sum_{m,n}\binom{t}{m,n,t-m-n}(1-p)^{(n+m)v}[1-(1-p)^v]^m[(1+p)^v-1]^n \times[1-(1-p)^v(1+p)^v]^{t-m-n}T(v,m,n). \tag{28}$$

This formula applies to both Probe Order Backtracking and Backtracking with Probing so long as the corresponding definition for T is used.

7.1.4 Heuristic Analysis

Before continuing with the exact analysis of the two algorithms we will give a brief heuristic analysis for the average time used by Probe Order Backtracking.

Ignore the fact that setting variables has an effect on clauses other than the selected clause. In particular, ignore the fact that the nonselected clauses can become empty or satisfied and ignore the fact that once the variables of one clause are set, there could be fewer variables waiting to be set in the remaining clauses. Under this radical assumption, the number of subproblems produced by splitting on the variables of the selected clause is the same as the length of the selected clause. Eq. (17) implies $T(v,m,n)$ is given by

$$2\left[\left(\frac{pv}{1-(1-p)^v}\right)^{m+1}-1\right]\Bigg/\left[\left(\frac{pv}{1-(1-p)^v}\right)-1\right]-1. \tag{29}$$

Details: If each of m clauses contains w variables, the total number of non-root nodes in the implied search tree satisfies the recurrence

$$N(m)=wN(m-1)+2w.$$

'$2w$' is the number of nodes arising from setting each variable in the clause *true* and *false* as specified in the Probe Order Backtracking Algorithm. There are w subproblems produced by splitting. Each subproblem has $m-1$ clauses.

The solution to this recurrence is

$$2\left[\frac{w^{m+1}-1}{w-1}\right]-2.$$

Add 1 for the root node and use eq. (17) for w to obtain eq. (29).$\diamond$

Plugging into eq. (28) and summing over m and n gives an average number of nodes of

$$1+\left(\frac{2pv}{pv-1+(1-p)^v}\right)\{[1+(pv-1)(1-p)^v]^t-[1-(1-p)^{2v}]^t\}. \tag{30}$$

Details: Define

$$w=\frac{pv}{1-(1-p)^v}.$$

Then

$$
\begin{aligned}
&1 - [1-(1-p)^{2v}]^t + \sum_{m,n} \binom{t}{m,n,t-m-n} (1-p)^{(n+m)v} [1-(1-p)^v]^m [(1+p)^v - 1]^n \\
&\quad \times [1-(1-p)^v(1+p)^v]^{t-m-n} \left\{ 2 \left[\frac{w^{m+1}-1}{w-1} \right] - 1 \right\} \\
&= 1 - 2[1-(1-p)^{2v}]^t + \left(\frac{2}{w-1} \right) \sum_m \binom{t}{m} (1-p)^{mv} [1-(1-p)^v]^m [1-(1-p)^v]^{t-m} \left\{ w^{m+1} - 1 \right\} \\
&= 1 - \left(\frac{2w}{w-1} \right) [1-(1-p)^{2v}]^t + \left(\frac{2w}{w-1} \right) \sum_m \binom{t}{m} [pv(1-p)^v]^m [1-(1-p)^v]^{t-m} \\
&= 1 - \left(\frac{2w}{w-1} \right) [1-(1-p)^{2v}]^t + \left(\frac{2w}{w-1} \right) [1+(pv-1)(1-p)^v]^t \\
&= 1 - \left(\frac{2pv}{pv-1+(1-p)^v} \right) [1-(1-p)^{2v}]^t + \left(\frac{2pv}{pv-1+(1-p)^v} \right) [1+(pv-1)(1-p)^v]^t.
\end{aligned}
$$

◇

The contours for this function are given in Fig. 4. Carefully comparing with the true answer (Fig. 2) we see that the heuristic analysis gives neither an upper bound nor a lower bound. For high p the values are too small (because changes in clause types were neglected) and for low p the values are too low (because the fact that nonselected clauses can become empty was neglected). This type of analysis can be useful during initial algorithm design because it is simple to do, and it often gives roughly the right answer. One must beware, however, that on some problems a similar approach might give a radically wrong answer.

7.1.5 Transition Probabilities

Suppose a predicate is produced by repeatedly generating random clauses from v variables. Suppose the resulting predicate contains m all positive clauses, n other clauses, and no empty clauses.

Let $G(v,n)$ be the probability that setting a random variable to *true* results in the predicate having one or more empty clauses. When a variable is set to *true*, other clauses become empty with the probability given in eq. (20) while all positive clauses do not become empty. Therefore,

$$G(v,n) = 1 - \left(1 - \frac{p(1-p)^{v-1}}{(1+p)^v - 1} \right)^n. \tag{31}$$

Let $F(v,m)$ be the probability that setting a random variable to *false* results in a predicate with one or more empty clauses. Eq. (15) implies

$$F(v,m) = 1 - \left(1 - \frac{p(1-p)^{v-1}}{1-(1-p)^v} \right)^m. \tag{32}$$

Let $D(v,i,k,m,n)$ be the probability that setting i random variables to *false* results in no clauses becoming empty and k other clauses becoming satisfied. If $i = 0$, nothing happens, so

$$D(v,0,k,m,n) = \delta_{k0}. \tag{33}$$

For $i = 1$, eqs. (15, 24) imply

$$D(v,1,k,m,n) = \binom{n}{k} \left(1 - \frac{p(1-p)^{v-1}}{1-(1-p)^v} \right)^m \left(\frac{p(1+p)^{v-1}}{(1+p)^v - 1} \right)^k \left(1 - \frac{p(1+p)^{v-1}}{(1+p)^v - 1} \right)^{n-k}. \tag{34}$$

For $i > 1$, some other clauses (x) must be satisfied when the first $i-1$ variables are set and then the rest ($k-x$) must be satisfied when the last variable is set, so D can be calculated from

$$D(v,i,k,m,n) = \sum_x D(v,i-1,x,m,n) D(v-i+1,1,k-x,m,n-x). \tag{35}$$

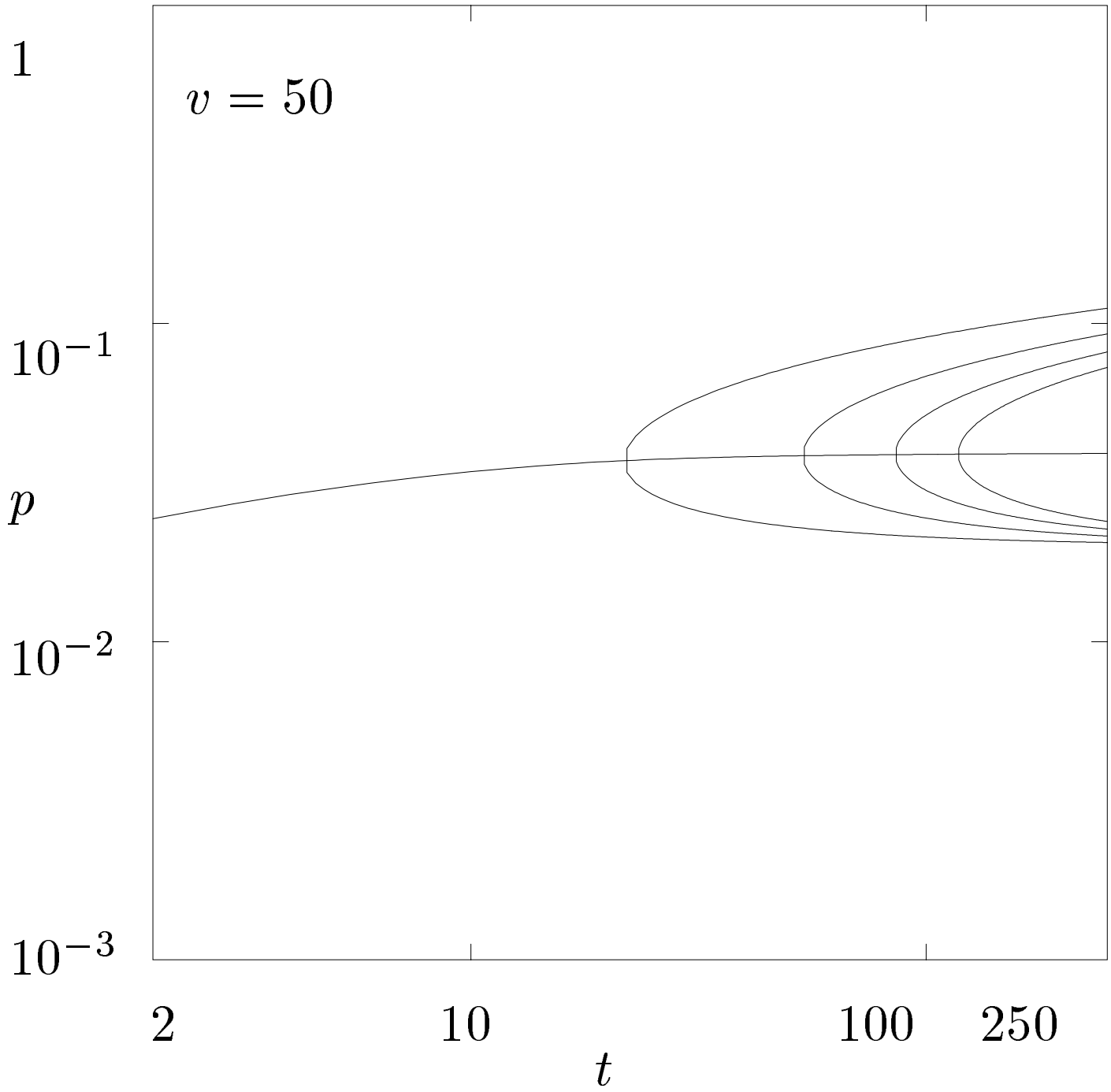

Fig. 4. Heuristic Analysis of Probe Order Backtracking.

Since the m index is constant in this recurrence and

$$\prod_{0 \le j < i} \left(1 - \frac{p(1-p)^{v-j-1}}{1-(1-p)^{v-j}}\right) = \frac{1-(1-p)^{v-i}}{1-(1-p)^{v}}, \tag{36}$$

we have

$$D(v,i,k,m,n) = \left(\frac{1-(1-p)^{v-i}}{1-(1-p)^{v}}\right)^{m} D(v,i,k,n), \tag{37}$$

where

$$D(v,1,k,n) = \binom{n}{k} \left(\frac{p(1+p)^{v-1}}{(1+p)^v - 1}\right)^k \left(1 - \frac{p(1+p)^{v-1}}{(1+p)^v - 1}\right)^{n-k}, \tag{38}$$

and

$$D(v,i,k,n) = \sum_x D(v,i-1,x,n) D(v-i+1,1,k-x,n-x). \tag{39}$$

By examining small cases, one finds

$$D(v,i,k,n) = \binom{n}{k} \frac{[(1+p)^v - (1+p)^{v-i}]^k [(1+p)^{v-i} - 1]^{n-k}}{[(1+p)^v - 1]^n}, \tag{40}$$

which can be proved by induction.

Details: For $i = 1$ eq. (40) reads

$$D(v,1,k,n) = \binom{n}{k} \frac{[(1+p)^v - (1+p)^{v-1}]^k [(1+p)^{v-1} - 1]^{n-k}}{[(1+p)^v - 1]^n},$$

which simplifies to eq. (38). Suppose eq. (39) is true for $i = 1$ and for $i = i_* - 1$. Then eq. (39) reads

$$\begin{aligned} D(v,i_*,k,n) &= \sum_x \binom{n}{x} \frac{[(1+p)^v - (1+p)^{v-i_*+1}]^x [(1+p)^{v-i_*+1} - 1]^{n-x}}{[(1+p)^v - 1]^n} \\ &\quad \times \binom{n-x}{k-x} \left(\frac{p(1+p)^{v-i_*}}{(1+p)^{v-i_*+1} - 1}\right)^{k-x} \left(1 - \frac{p(1+p)^{v-i_*}}{(1+p)^{v-i_*+1} - 1}\right)^{n-k} \\ &= \binom{n}{k} [(1+p)^{v-i_*} - 1]^{n-k} [(1+p)^v - 1]^{-n} \\ &\quad \times \sum_x \binom{k}{x} [(1+p)^v - (1+p)^{v-i_*+1}]^x [p(1+p)^{v-i_*}]^{k-x}, \end{aligned}$$

which simplifies to eq. (40). $\diamond$

Let $E(v,j,k,l,m,n)$ be the probability that setting one random variable to *true* results in no clauses becoming empty, j other clauses becoming all positive clauses, k other clauses becoming satisfied, and l all positive clauses becoming satisfied. (Notice that no other changes of clause category can occur.) Eqs. (13, 20, 21, 22) imply

$$\begin{aligned} E(v,j,k,l,m,n) &= \binom{m}{l} \binom{n}{j,k,n-j-k} \left(\frac{p}{1-(1-p)^v}\right)^l \left(1 - \frac{p}{1-(1-p)^v}\right)^{m-l} \left(\frac{p[1-(1-p)^{v-1}]}{(1+p)^v - 1}\right)^j \\ &\quad \times \left(\frac{p[(1+p)^{v-1} - 1]}{(1+p)^v - 1}\right)^k \left(1 - \frac{p(1+p)^{v-1}}{(1+p)^v - 1}\right)^{n-j-k}. \end{aligned} \tag{41}$$

Details: From eq. (21) the probability that an other clause will become all positive is

$$\sum_{j \geq 1} \binom{v-1}{j} \frac{p^{j+1}(1-p)^{v-j-1}}{(1+p)^v - 1} = \frac{p[1-(1-p)^{v-1}]}{(1+p)^v - 1}.$$

$$\begin{aligned} E(v,j,k,l,m,n) &= \binom{m}{l} \binom{n}{j,k,n-j-k} \left(\frac{p}{1-(1-p)^v}\right)^l \left(1 - \frac{p}{1-(1-p)^v}\right)^{m-l} \left(\frac{p[1-(1-p)^{v-1}]}{(1+p)^v - 1}\right)^j \\ &\quad \times \left(\frac{p[(1+p)^{v-1} - 1]}{(1+p)^v - 1}\right)^k \\ &\quad \times \left(1 - \frac{p(1-p)^{v-1}}{(1+p)^v - 1} - \frac{p[1-(1-p)^{v-1}]}{(1+p)^v - 1} - \frac{p[(1+p)^{v-1} - 1]}{(1+p)^v - 1}\right)^{n-j-k} \\ &= \binom{m}{l} \binom{n}{j,k,n-j-k} \left(\frac{p}{1-(1-p)^v}\right)^l \left(1 - \frac{p}{1-(1-p)^v}\right)^{m-l} \left(\frac{p[1-(1-p)^{v-1}]}{(1+p)^v - 1}\right)^j \\ &\quad \times \left(\frac{p[(1+p)^{v-1} - 1]}{(1+p)^v - 1}\right)^k \left(1 - \frac{p(1+p)^{v-1}}{(1+p)^v - 1}\right)^{n-j-k}. \end{aligned}$$

$\diamond$

7.2 Backtracking with Probing

Let $T(v, m, n)$ be the average number of nodes for a problem solved by the Backtracking with Probing Algorithm that has v variables, m all positive clauses, and n other clauses, and no empty clauses. If m or n is zero, then the algorithm stops immediately, so there is only one node. Thus,

$$T(v, 0, n) = T(v, m, 0) = 1. \tag{42}$$

If both m and n are bigger than zero, then there are some nodes for the subtree that results when the selected variable is set to *false*, some nodes for the subtree that results when the selected variable is set to *true*, and one node for the root of the search tree.

When the variable is set to *false*, with probability $F(v, m)$ an empty clause is produced (and therefore there is one node in the subtree). With probability $D(v, 1, k, m, n)$, no empty clauses are produced and k of the other clauses become satisfied, resulting in $T(v-1, m, n-k)$ as the expected number of nodes in the subtree.

When the variable is set to *true*, with probability $G(v, n)$ an empty clause is produced. With probability $E(v, j, k, l, m, n)$, no empty clauses are produced, j other clauses are converted into all positive clauses, k other clauses are satisfied, and l all positive clauses are satisfied, resulting in $T(v-1, m+j-l, n-k-j)$ as the expected number of nodes in the subtree.

For Backtracking with Probing, adding up the nodes from all the cases gives

$$\begin{aligned} T(v, m, n) = 1 + F(v, m) + G(v, n) + \sum_{k} D(v, 1, k, m, n) T(v-1, m, n-k) \\ + \sum_{j,k,l} E(v, j, k, l, m, n) T(v-1, m+j-l, n-k-j). \end{aligned} \tag{43}$$

7.3 Probe Order Backtracking

Probe Order Backtracking selects a clause and then sets the variables that occur in the clause. If the selected clause has h variables, then there is a root, a node from setting the first variable to *false*, a potential node from setting the first two variables to *false*, and so on. This gives a root plus up to h additional nodes. In addition, there is a subtree for setting the first variable to *true*, potentially a subtree for setting the first variable to *false* and the second to *true*, and so on. When setting the first few variables, some of the other clauses may evaluate to *false*. Also, setting the first few variables may result in the number of other clauses dropping to zero. Either of these effects may prevent a potential node from occurring in the tree.

Define $a(v, i)$ as the probability that the selected clause contains i or more nodes (thus potentially contributing an i^{th} node to the backtrack tree). Then, from eq. (12) we obtain

$$a(v, i) = \sum_{j \geq i} A(v, j) = \sum_{j \geq i} \binom{v}{j} \frac{p^j (1-p)^{v-j}}{1-(1-p)^v}. \tag{44}$$

Let $T(v, m, n)$ be the average number of nodes for a problem that has v variables, m all positive clauses, n other clauses and no empty clauses. For Probe Order Backtracking

$$\begin{aligned} T(v, m, n) = 1 + \sum_{1 \leq i \leq v} a(v, i) \sum_{x < n} D(v, i-1, x, m-1, n) \Big[1 + G(v-i+1, n-x) \\ + \sum_{j,k,l} E(v-i+1, j, k, l, m-1, n-x) T(v-i, m+j-l-1, n-j-k-x) \Big]. \end{aligned} \tag{45}$$

The initial 1 is for the root of the tree. The i index is for those nodes that occur as a result of setting the first i variables from the clause. The factor $a(v, i)$ gives the probability that the selected clause has at least i variables. The index x is for the number of other clauses that are satisfied when setting the first $i-1$ variables *false*. The sum does not include $x = n$, because no subproblems are generated when the number

of other clauses is reduced to zero. The factor $D(v, i-1, x, m-1, n)$ is the probability that x of the n other clauses become satisfied and no clauses become empty as a result of setting the first $i-1$ variables. The D factor multiplies the sum of terms that relate to the various kinds of nodes that can result when the i^{th} variable is set. The 1 following the square bracket is for the node that results from setting the i^{th} variable to *false*. The $G(v-i+1, n-x)$ term gives the probability that setting the i^{th} variable to *true* produces an empty clause. When setting the i^{th} variable to *true*, the j index counts the number of other clauses that become all positive, the k index counts the number of other clauses that become satisfied, and the l index counts the number of all positive clauses that are satisfied (the selected clause is not included in this count). The factor $E(v-i+1, j, k, l, m-1, n-x)$ is the probability that setting the i^{th} variable results in the values j, k, and l. The factor $T(v-i, m+j-l-1, n-j-k-x)$ is the expected number of nodes in the subtree that results from setting the first $i-1$ variables to *false* and the i^{th} variable to *true*.

As with the previous analysis, the boundary conditions are.

$$T(v, 0, n) = T(v, m, 0) = 1. \tag{46}$$

After a number of algebraic transformations eq. (45) can be rewritten as

$$T(v, m, n) = 1 + \sum_{1 \le i \le v} a(v, i) \left(Z(v, i, m, n) + \sum_{j,k} H(v, i, j, k, m, n) T(v-i, j, k) \right), \tag{47}$$

where

$$\begin{aligned} Z(v, i, m, n) = & \left(\frac{1-(1-p)^{v-i+1}}{1-(1-p)^v} \right)^{m-1} \\ & \times \left[2 - \left(\frac{(1+p)^v - (1+p)^{v-i+1}}{(1+p)^v - 1} \right)^n - \left(\frac{(1+p)^v - 1 - p(1-p)^{v-i}}{(1+p)^v - 1} \right)^n \right], \end{aligned} \tag{48}$$

and

$$\begin{aligned} H(v, i, j, k, m, n) = & \left(\frac{(1+p)^{v-i} - 1}{(1+p)^v - 1} \right)^k \sum_l \binom{m-1}{l} \binom{n}{l+j-m+1} \binom{n+m-l-j-1}{k} \\ & \times \left(\frac{p}{1-(1-p)^v} \right)^l \left(\frac{1-(1-p)^{v-i+1} - p}{1-(1-p)^v} \right)^{m-l-1} \left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^v - 1} \right)^{l+j-m+1} \\ & \times \left(\frac{(1+p)^v - (1+p)^{v-i} - p}{(1+p)^v - 1} \right)^{n+m-l-j-k-1} \\ & - \delta_{k0} \left(\frac{(1+p)^v - (1+p)^{v-i+1}}{(1+p)^v - 1} \right)^n \\ & \times \binom{m-1}{j} \left(\frac{1-(1-p)^{v-i+1} - p}{1-(1-p)^v} \right)^j \left(\frac{p}{1-(1-p)^v} \right)^{m-1-j}. \end{aligned} \tag{49}$$

Details: Define

$$Y(v, i) = \frac{1-(1-p)^{v-i}}{1-(1-p)^v}.$$

Then eqs. (37, 45) imply

$$\begin{aligned} T(v, m, n) = 1 + \sum_{1 \le i \le v} a(v, i) \sum_{x < n} [Y(v, i-1)]^{m-1} D(v, i-1, x, n) \Big[& 1 + G(v-i+1, 1, n-x) \\ & + \sum_j \sum_k \sum_l E(v-i+1, j, k, l, m-1, n-x) T(v-i, m+j-l-1, n-j-k-x) \Big]. \end{aligned}$$

$\diamond$

The key idea in the derivation is to first change indices with $j' = m + j - l - 1$ and $k' = n - j - k - x$.

Details: Thus, $j = l + j' - m + 1$, $k = n + m - l - x - j' - k' - 1$, and

$$T(v,m,n) = 1 + \sum_{1 \le i \le v} a(v,i) \sum_{x<n} [Y(v,i-1)]^{m-1} D(v,i-1,x,n) \Bigg[1 + G(v-i+1,1,n-x) + \sum_j \sum_k \sum_l E(v-i+1, l+j-m+1, n+m-l-x-j-k-1, l, m-1, n-x) T(v-i,j,k) \Bigg].$$

◇

Then one computes the sum over x and names the coefficients to obtain eq. (47).

Details: Define

$$Z(v,i,m,n) = \sum_{x<n} [Y(v,i-1)]^{m-1} D(v,i-1,x,n)[1 + G(v-i+1,1,n-x)].$$

$$\begin{aligned} Z(v,i,m,n) &= [Y(v,i-1)]^{m-1} \sum_x \binom{n}{x} \frac{[(1+p)^v - (1+p)^{v-i+1}]^x [(1+p)^{v-i+1} - 1]^{n-x}}{[(1+p)^v - 1]^n} (1 - \delta_{xn}) \\ &\quad \times \left[2 - \left(1 - \frac{p(1-p)^{v-i}}{(1+p)^{v-i+1} - 1} \right)^{n-x} \right] \\ &= [Y(v,i-1)]^{m-1} \left[2 - \left(\frac{(1+p)^v - (1+p)^{v-i+1}}{(1+p)^v - 1} \right)^n - \left(\frac{(1+p)^v - 1 - p(1-p)^{v-i}}{(1+p)^v - 1} \right)^n \right]. \end{aligned}$$

Define

$$\begin{aligned} H(v,i,j,k,m,n) &= \sum_{x<n} [Y(v,i-1)]^{m-1} D(v,i-1,x,n) \\ &\quad \sum_l E(v-i+1, l+j-m+1, n+m-l-x-j-k-1, l, m-1, n-x) \\ &= [Y(v,i-1)]^{m-1} \sum_x \binom{n}{x} \frac{[(1+p)^v - (1+p)^{v-i+1}]^x [(1+p)^{v-i+1} - 1]^{n-x}}{[(1+p)^v - 1]^n} (1 - \delta_{xn}) \\ &\quad \sum_l \binom{m-1}{l} \binom{n-x}{l+j-m+1, n+m-l-x-j-k-1, k} \left(\frac{p}{1-(1-p)^{v-i+1}} \right)^l \\ &\quad \times \left(1 - \frac{p}{1-(1-p)^{v-i+1}} \right)^{m-l-1} \left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^{v-i+1} - 1} \right)^{l+j-m+1} \\ &\quad \times \left(\frac{p[(1+p)^{v-i} - 1]}{(1+p)^{v-i+1} - 1} \right)^{n+m-l-x-j-k-1} \left(1 - \frac{p(1+p)^{v-i}}{(1+p)^{v-i+1} - 1} \right)^k, \end{aligned}$$

$$
\begin{aligned}
H(v,i,j,k,m,n) &= [Y(v,i-1)]^{m-1} \sum_x \frac{[(1+p)^v-(1+p)^{v-i+1}]^x[(1+p)^{v-i+1}-1]^{n-x}}{[(1+p)^v-1]^n}(1-\delta_{xn}) \\
&\quad \sum_l \binom{m-1}{l}\binom{n}{l+j-m+1}\binom{n+m-l-j-1}{k}\binom{n+m-l-j-k-1}{x} \\
&\quad\quad \times \left(\frac{p}{1-(1-p)^{v-i+1}}\right)^l \left(1-\frac{p}{1-(1-p)^{v-i+1}}\right)^{m-l-1} \\
&\quad\quad \times \left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^{v-i+1}-1}\right)^{l+j-m+1} \left(\frac{p[(1+p)^{v-i}-1]}{(1+p)^{v-i+1}-1}\right)^{n+m-l-x-j-k-1} \\
&\quad\quad \times \left(1-\frac{p(1+p)^{v-i}}{(1+p)^{v-i+1}-1}\right)^k \\
&= [Y(v,i-1)]^{m-1}\left(\frac{(1+p)^{v-i+1}-1}{(1+p)^v-1}\right)^n \\
&\quad \sum_l \binom{m-1}{l}\binom{n}{l+j-m+1}\binom{n+m-l-j-1}{k}\left(\frac{p}{1-(1-p)^{v-i+1}}\right)^l \\
&\quad\quad \times \left(1-\frac{p}{1-(1-p)^{v-i+1}}\right)^{m-l-1}\left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^{v-i+1}-1}\right)^{l+j-m+1}\left(1-\frac{p(1+p)^{v-i}}{(1+p)^{v-i+1}-1}\right)^k \\
&\quad\quad \times \sum_x \binom{n+m-l-j-k-1}{x}\left(\frac{(1+p)^v-(1+p)^{v-i+1}}{(1+p)^{v-i+1}-1}\right)^x \\
&\quad\quad\quad \times \left(\frac{p[(1+p)^{v-i}-1]}{(1+p)^{v-i+1}-1}\right)^{n+m-l-x-j-k-1}(1-\delta_{xn}) \\
&= [Y(v,i-1)]^{m-1}\left(\frac{(1+p)^{v-i+1}-1}{(1+p)^v-1}\right)^n \\
&\quad \sum_l \binom{m-1}{l}\binom{n}{l+j-m+1}\binom{n+m-l-j-1}{k}\left(\frac{p}{1-(1-p)^{v-i+1}}\right)^l \\
&\quad\quad \times \left(1-\frac{p}{1-(1-p)^{v-i+1}}\right)^{m-l-1}\left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^{v-i+1}-1}\right)^{l+j-m+1}\left(\frac{(1+p)^{v-i}-1}{(1+p)^{v-i+1}-1}\right)^k \\
&\quad\quad \times \left[\left(\frac{(1+p)^v-(1+p)^{v-i}-p}{(1+p)^{v-i+1}-1}\right)^{n+m-l-j-k-1}\right. \\
&\quad\quad\quad \left. -\binom{n+m-l-j-k-1}{n}\left(\frac{(1+p)^v-(1+p)^{v-i+1}}{(1+p)^{v-i+1}-1}\right)^n\left(\frac{p[(1+p)^{v-i}-1]}{(1+p)^{v-i+1}-1}\right)^{m-l-j-k-1}\right].
\end{aligned}
$$

Cancel factors to obtain eq. (49).$\diamond$

7.4 Alternate Recurrence

Time $O(v^2t^4)$ is needed to calculate the average number of nodes for Probe Order Backtracking from eqs. (28, 47). The following equations calculate the same results in time $O(vt^4+v^2t^3)$.

$$
\begin{aligned}
T(v,m,n,i) &= 2[1-(1-p)^i](1-p)^{v-i}[(1-p)^v-(1-p)^{2v}]^{m-1}[(1-p^2)^v-(1-p)^v]^n \\
&\quad + [1-(1-p)^i](1-p)^{v-i}(1-p)^{n+m-1}(1-p)^{v-1}\sum_k \binom{n}{k}p^{n-k} \\
&\quad\quad \times \sum_j J(v-1,m-1,n-k,j)T(v-1,j,k,v-1) \\
&\quad + p(1-p)^{n+m-1}\sum_{0<f<i}\sum_k \binom{n}{k}[p(1-p^2)^{v-1}]^k T(v-1,m,n-k,f), \qquad (50)
\end{aligned}
$$

$$T(0,m,n,i) = T(v,0,n,i) = T(v,m,0,i) = T(v,m,n,0) = 0. \tag{51}$$

The factor $J(v,m,k,j)$ may be calculated from the recurrence

$$\begin{aligned} J(v,m,k,j) &= [(1-p^2)^v - (1-p)^v]J(v,m,k-1,j) + J(v,m,k-1,j-1) \\ J(v,m,0,j) &= \binom{m}{j}(1-p)^j[p(1-p)^v]^{m-j} \\ J(v,m,k,0) &= [p(1-p)^v]^m[(1-p^2)^v - (1-p)^v]^k. \end{aligned} \tag{52}$$

The expected number of nodes for a problem with v variables and t clauses solved by Probe Order Backtracking is

$$1 + \sum_m \sum_n \binom{t}{m,n,t-m-n}(1-p)^v[1-(1-p^2)^v]^{t-m-n}T(v,m,n,v). \tag{53}$$

In order to connect this version of the analysis to the analysis expressed in eqs. (28, 47) we define

$$, (v,m,n,i) = [1-(1-p)^i](1-p)^{v-i}[(1-p)^v - (1-p)^{2v}]^{m-1}[(1-p^2)^v - (1-p)^v]^n. \tag{54}$$

Then we may relate T from eq. (50) to T from eq. (47) via

$$T(v,m,n,v) = , (v,m,n,v)[T(v,m,n) - 1]. \tag{55}$$

$T(v,m,n,i)/, (v,m,n,i)$ is the expected number of non-root nodes in the backtrack tree generated by a problem with v variables, n other clauses, m all positive clauses, and no empty clauses such that the first all positive clause may not contain any of the first $v-i$ variables. The first three parameters have the same meaning as the corresponding parameters from eq. (47). The fourth parameter, i, keeps track of the effect setting literals *false* has in shortening the first all positive clause during the course of the algorithm. The factor , (v,m,n,i) is chosen so that no divisions are needed to evaluate eqs. (50–53).

Although eq. (50) uses four indices, it is a full-history recurrence in only three indices: m, n, and i. Hence $T(v,m,n,i)$ may be calculated in space $O(vt^2)$.

Details: Eqs. (50–53) are an analysis of the same algorithm as eqs. (28, 47). However, in deriving eqs. (50–53) it is clearest to restate the Probe Order Algorithm in an equivalent but more explicitly recursive form. Begin by establishing a canonical ordering of the literals and clauses in the predicate. Remove all tautological clauses. Then:

1. (**Test: Empty Predicate**) If the predicate is empty, return the solution.
2. (**Test: Empty Clause**) If any clause is empty, return with no solution.
3. (**Probe: Negatively-Augmented Solution**) If all remaining clauses have a negative literal, return the negatively-augmented solution. (That is, check for $m = 0$.)
4. (**Partial Probe: Positively-Augmented Solution**) If no remaining clauses have a negative literal, return the positively-augmented solution. (That is, check for $n = 0$.)
5. (**Splitting: Clause-Order Recursion**) Consider the first positive literal (according to the canonical ordering of the literals) appearing in the first all positive clause:
   a. (**Assert the Literal.**) Set the literal *true*. Simplify the predicate. Charge one time unit. Recur with the simplified predicate.
   b. (**Negate the Literal.**) Set the literal *false*. Simplify the predicate. Charge one time unit. Recur with the simplified predicate.

Conclude the algorithm by charging one time unit for the root.

We count the nodes in the backtrack tree by recursively counting the nodes introduced by each step of the algorithm. We must keep track of how many all positive clauses and how many other clauses remain at each stage of the algorithm. In addition we must keep track of events that affect the length of the first all positive clause.

Let $T_1(v,m,n,i)$ be the expected number of nodes (exclusive of the root) in the backtrack tree generated by a call to Probe Order Backtracking on a predicate with v variables, m all positive clauses, n other

clauses, and no empty clauses, in which the first all positive clause is drawn from the last i variables in the canonical listing of variables. We shall see that $T_1(v,m,n,i)$ is related to $T(v,m,n,i)$ from eq. (50) via $T(v,m,n,i) = , (v,m,n,i)T_1(v,m,n,i)$.

Let $Q(t,v)$ be the expected number of nodes in the backtrack tree generated by a call to Probe Order Backtracking on a predicate with t clauses and v variables. Then from eq. (28) we have

$$Q(t,v) = 1 + \sum_m \sum_n \binom{t}{m,n,t-m-n} [(1-p)^v - (1-p)^{2v}]^m \times [(1-p^2)^v - (1-p)^v]^n [1-(1-p^2)^v]^{t-m-n} T_1(v,m,n,v).$$

In degenerate cases the algorithm does not set any variables. Hence the natural boundary conditions for $T_1(v,m,n,i)$ are

$$T_1(0,m,n,i) = T_1(v,0,n,i) = T_1(v,m,0,i) = T_1(v,m,n,0) = 0.$$

Nodes may be introduced in one of two ways: the algorithm *Asserts the Literal*, or the algorithm *Negates the Literal*. Let us denote the expected number of nodes introduced by these branches as $A(v,m,n,i)$ and $N(v,m,n,i)$, respectively. Write

$$T_1(v,m,n,i) = A(v,m,n,i) + N(v,m,n,i).$$

We now determine the equations for $A(v,m,n,i)$ and $N(v,m,n,i)$. $\diamond$

Details: 7.4.1 Derivation of A

Suppose the first literal in the first all positive clause is asserted. One time unit is charged and the predicate is simplified. Probe Order Backtracking is called recursively on the simplified predicate.

Using eq. (41), the total number of nodes introduced by setting the literal *true* is given by

$$A(v,m,n,i) = 1 + \sum_j \sum_k \sum_l E(v,j,k,l,m-1,n) T_1(v-1, m-1+j-l, n-j-k, v-1).$$

Notice that $A(v,m,n,i)$ is independent of i. It is useful to define $A(v,m,n) = A(v,m+1,n,i)$ (notice the shift in the m index). $\diamond$

Details: 7.4.2 Derivation of N

Suppose the first literal in the first all positive clause is set *false*. One time unit is charged and the predicate is simplified. Probe Order Backtracking is called recursively on the simplified predicate.

The first all positive clause is drawn from a population of i variables (the last i variables in the canonical listing of variables). The probability that a random clause drawn from i variables contains no variable appearing negatively is $(1-p)^i$. The probability that none of the first $i-f-1$ variables occurs positively in the clause is $(1-p)^{i-f-1}$. The probability that this clause contains the positive form of the $(i-f)^{\text{th}}$ variable is p. The probability that at least one of the remaining f variables occurs positively is $1-(1-p)^f$. Eq. (11) gives the probability that a random clause is all positive. Combining all these factors, we obtain the probability, given a random all-positive clause in i variables, that the first literal to appear in this clause will be the positive form of the $(i-f)^{\text{th}}$ variable in the canonical listing of the i variables, and that after setting the $(i-f)^{\text{th}}$ literal *false* the clause is not empty:

$$\frac{(1-p)^i(1-p)^{i-f-1}p[1-(1-p)^f]}{(1-p)^i - (1-p)^{2i}} = \frac{p(1-p)^{i-f-1} - p(1-p)^{i-1}}{1-(1-p)^i}.$$

Stated another way, this is the probability that setting the first literal *false* in the first all positive clause will result in a subproblem in which the first all positive clause is drawn from a population of f variables.

From eq. (34) and the preceding discussion, setting the literal *false* leaves a problem in $v-1$ variables, m all positive clauses, $n-k$ other clauses, and no empty clauses, in which the first all positive clause is drawn from the last f variables with probability

$$\left[\frac{p(1-p)^{i-f-1} - p(1-p)^{i-1}}{1-(1-p)^i}\right] D(v,1,k,m-1,n).$$

Summing over all cases, the total number of nodes introduced by setting the literal *false* is given by

$$N(v,m,n,i) = 1 + \sum_k \sum_{0<f<i} \left[\frac{p(1-p)^{i-f-1} - p(1-p)^{i-1}}{1-(1-p)^i}\right] D(v,1,k,m-1,n) T_1(v-1,m,n-k,f).$$

$\diamond$

Details: 7.4.3 Full Set of Equations for the Alternate Recurrence

We now simplify the following set of equations:

$$Q(t,v) = 1 + \sum_{0<m<t} \sum_{0<n\le t-m} \binom{t}{m,n,t-m-n} [(1-p)^v - (1-p)^{2v}]^m \times [(1-p^2)^v - (1-p)^v]^n [1-(1-p^2)^v]^{t-m-n} T_1(v,m,n,v),$$

$$T_1(v,m,n,i) = A(v,m-1,n) + N(v,m,n,i),$$

$$T_1(0,m,n,i) = T_1(v,0,n,i) = T_1(v,m,0,i) = T_1(v,m,n,0) = 0,$$

$$A(v,m,n) = 1 + \sum_j \sum_k \sum_l E(v,j,k,l,m,n) T_1(v-1, m+j-l, n-j-k, v-1),$$

$$N(v,m,n,i) = 1 + \sum_k \sum_{0<f<i} \left[\frac{p(1-p)^{i-f-1} - p(1-p)^{i-1}}{1-(1-p)^i} \right] D(v,1,k,m-1,n) T_1(v-1,m,n-k,f).$$

Perform the index changes $j' = m+j-l$ and $k' = n-k-j$ in $A(v,m,n)$.

$$A(v,m,n) = 1 + \sum_{j,k,l} E(v, j-m+l, n+m-j-k-l, l, m, n) T_1(v-1, j, k, v-1).$$

Plug in the definition for E from eq. (41).

$$A(v,m,n) = 1 + \sum_{j,k,l} \binom{m}{l} \binom{n}{j-m+l, n+m-j-k-l, k} \left(\frac{p}{1-(1-p)^v} \right)^l \times \left(1 - \frac{p}{1-(1-p)^v} \right)^{m-l} \left(\frac{p[1-(1-p)^{v-1}]}{(1+p)^v - 1} \right)^{j-m+l} \left(\frac{p[(1+p)^{v-1}-1]}{(1+p)^v-1} \right)^{n+m-j-k-l} \times \left(1 - \frac{p(1+p)^{v-1}}{(1+p)^v-1} \right)^k T_1(v-1,j,k,v-1).$$

Clear some fractions to yield:

$$A(v,m,n) = 1 + \left[\frac{(1-p)^v}{(1-p)^v[(1+p)^v-1]} \right]^n \left[\frac{(1-p)^v}{(1-p)^v[1-(1-p)^v]} \right]^m \sum_{j,k,l} \binom{m}{l} \binom{n}{k} \binom{n-k}{j-(m-l)} \times p^l (1-p)^{m-l} p^{n-k} [1-(1-p)^{v-1}]^j [(1+p)^{v-1}-1]^{n+m-j-l} T_1(v-1,j,k,v-1).$$

In order to guard against floating point overflow due to the factor of $(1+p)^v - 1$ we have introduced a factor of $(1-p)^{v(m+n)}/(1-p)^{v(m+n)}$ into the summation. Multiplying this through gives

$$A(v,m,n) = 1 + (1-p)^{n+m} \left[\frac{1}{(1-p^2)^v - (1-p)^v} \right]^n \left[\frac{1}{(1-p)^v - (1-p)^{2v}} \right]^m \sum_{j,k,l} \binom{m}{l} \binom{n}{k} \binom{n-k}{j-(m-l)} \times p^l (1-p)^{m-l} p^{n-k} [(1-p)^{v-1} - (1-p)^{2v-2}]^j [(1-p^2)^{v-1} - (1-p)^{v-1}]^k [(1-p)^{v-1}]^l \times [(1-p^2)^{v-1} - (1-p)^{v-1}]^{n-k-j+(m-l)} T_1(v-1,j,k,v-1).$$

Collect all the terms that depend on l and define

$$J(v,m,k,j) = \sum_l \binom{m}{l} \binom{k}{j-(m-l)} (1-p)^{m-l} [p(1-p)^v]^l [(1-p^2)^v - (1-p)^v]^{k-j+(m-l)},$$

then, after rearranging to emphasize speed of computation, we may write

$$A(v,m,n) = 1 + (1-p)^{n+m} \left[\frac{1}{(1-p^2)^v - (1-p)^v} \right]^n \left[\frac{1}{(1-p)^v - (1-p)^{2v}} \right]^m \times \sum_k \binom{n}{k} p^{n-k} [(1-p^2)^{v-1} - (1-p)^{v-1}]^k \times \sum_j [(1-p)^{v-1} - (1-p)^{2v-2}]^j J(v-1,m,n-k,j) T_1(v-1,j,k,v-1).$$

We avoid explicitly performing the sum over l in the evaluation of $J(v,m,k,j)$ by using a recurrence for $J(v,m,k,j)$. Using a recurrence for the binomial coefficient we have

$$\begin{aligned}
J(v,m,k,j) &= \sum_l \binom{m}{l}\binom{k-1}{j-(m-l)}(1-p)^{m-l}[p(1-p)^v]^l[(1-p^2)^v-(1-p)^v]^{k-j+(m-l)} \\
&\quad + \sum_l \binom{m}{l}\binom{k-1}{(j-1)-(m-l)}(1-p)^{m-l}[p(1-p)^v]^l[(1-p^2)^v-(1-p)^v]^{k-j+(m-l)} \\
&= [(1-p^2)^v-(1-p)^v] \\
&\quad \times \sum_l \binom{m}{l}\binom{k-1}{j-(m-l)}(1-p)^{m-l}[p(1-p)^v]^l[(1-p^2)^v-(1-p)^v]^{k-1-j+(m-l)} \\
&\quad + \sum_l \binom{m}{l}\binom{k-1}{(j-1)-(m-l)}(1-p)^{m-l}[p(1-p)^v]^l[(1-p^2)^v-(1-p)^v]^{k-1-(j-1)+(m-l)} \\
&= [(1-p^2)^v-(1-p)^v]J(v,m,k-1,j)+J(v,m,k-1,j-1).
\end{aligned}$$

To get the boundary conditions in eq. (52) note that the sum for $J(v,m,k,j)$ can be evaluated directly when $k=0$ or $j=0$.

$$\begin{aligned}
J(v,m,0,j) &= \binom{m}{j}(1-p)^j[p(1-p)^v]^{m-j} \\
J(v,m,k,0) &= [p(1-p)^v]^m[(1-p^2)^v-(1-p)^v]^k.
\end{aligned}$$

Now we work on $N(v,m,n,i)$. Plug the definition for $D(v,1,k,m-1,n)$ from eq. (34) into $N(v,m,n,i)$.

$$\begin{aligned}
N(v,m,n,i) = 1 + \sum_k \sum_{0<f<i} &\left[\frac{p(1-p)^{i-f-1}-p(1-p)^{i-1}}{1-(1-p)^i}\right]\binom{n}{k}\left(1-\frac{p(1-p)^{v-1}}{1-(1-p)^v}\right)^{m-1} \\
&\times \left(\frac{p(1+p)^{v-1}}{(1+p)^v-1}\right)^k\left(1-\frac{p(1+p)^{v-1}}{(1+p)^v-1}\right)^{n-k} T_1(v-1,m,n-k,f).
\end{aligned}$$

Clearing fractions in $N(v,m,n,i)$ gives:

$$\begin{aligned}
N(v,m,n,i) = 1 + p &\left(\frac{(1-p)^v}{(1-p)^v[(1+p)^v-1]}\right)^n\left(\frac{(1-p)^v}{(1-p)^v[1-(1-p)^v]}\right)^{m-1}\left[\frac{(1-p)^{i-1}}{1-(1-p)^i}\right] \\
&\times \sum_k \sum_{0<f<i}\left[\frac{1-(1-p)^f}{(1-p)^f}\right]\binom{n}{k}[1-(1-p)^{v-1}]^{m-1} \\
&\times [p(1+p)^{v-1}]^k[(1+p)^{v-1}-1]^{n-k}T_1(v-1,m,n-k,f).
\end{aligned}$$

Again, in order to guard against floating point overflow due to the factor of $(1+p)^v-1$, we have introduced a factor of $(1-p)^{v(m-1+n)}/(1-p)^{v(m-1+n)}$ into the summation. Multiplying this through and rearranging gives

$$\begin{aligned}
N(v,m,n,i) = 1 + p(1-p)^{n+m-1} &\left[\frac{1}{(1-p^2)^v-(1-p)^v}\right]^n\left[\frac{1}{(1-p)^v-(1-p)^{2v}}\right]^{m-1}\left[\frac{(1-p)^{i-1}}{1-(1-p)^i}\right] \\
&\times \sum_{0<f<i}\sum_k \binom{n}{k}[p(1-p^2)^{v-1}]^k[(1-p^2)^{v-1}-(1-p)^{v-1}]^{n-k} \\
&\times [(1-p)^{v-1}-(1-p)^{2v-2}]^{m-1}\left[\frac{1-(1-p)^f}{(1-p)^f}\right]T_1(v-1,m,n-k,f).
\end{aligned}$$

Now plug the definitions for $A(v, m-1, n)$ and $N(v, m, n, i)$ into $T_1(v, m, n, i)$.

$$
\begin{aligned}
T_1(v,m,n,i) = 2 &+ (1-p)^{n+m-1}\left[\frac{1}{(1-p^2)^v-(1-p)^v}\right]^n\left[\frac{1}{(1-p)^v-(1-p)^{2v}}\right]^{m-1} \\
&\times \sum_k \binom{n}{k} p^{n-k}[(1-p^2)^{v-1}-(1-p)^{v-1}]^k \\
&\times \sum_j [(1-p)^{v-1}-(1-p)^{2v-2}]^j J(v-1,m-1,n-k,j) T_1(v-1,j,k,v-1) \\
&+ p(1-p)^{n+m-1}\left[\frac{1}{(1-p^2)^v-(1-p)^v}\right]^n\left[\frac{1}{(1-p)^v-(1-p)^{2v}}\right]^{m-1}\left[\frac{(1-p)^{i-1}}{1-(1-p)^i}\right] \\
&\times \sum_{0<f<i}\sum_k \binom{n}{k}[p(1-p^2)^{v-1}]^k[(1-p^2)^{v-1}-(1-p)^{v-1}]^{n-k} \\
&\times [(1-p)^{v-1}-(1-p)^{2v-2}]^{m-1}\left[\frac{1-(1-p)^f}{(1-p)^f}\right] T_1(v-1,m,n-k,f).
\end{aligned}
$$

To clear the rest of the fractions we redefine $T_1(v, m, n, i)$:

$$
\begin{aligned}
T(v,m,n,i) &= [1-(1-p)^i](1-p)^{v-i}[(1-p)^v-(1-p)^{2v}]^{m-1}[(1-p^2)^v-(1-p)^v]^n T_1(v,m,n,i) \\
&= , (v,m,n,i) T_1(v,m,n,i).
\end{aligned}
$$

Making this substitution yields equations (50) and (53).◇

Details: 7.5 Verification of the Recurrences

Aside from being careful with the mathematics, we performed measurements to help insure the correctness of the analyses of Backtracking with Probing and of Probe Order Backtracking.

For each algorithm and for t and v in the range $1 \le v \le 6$, $1 \le t \le 6$, $1 \le tv \le 12$, we generated each of the 2^{2tv} SAT problems and counted the number of nodes produced. A problem with i literals has probability $p^i(1-p)^{2tv-i}$. Multiplying the node counts for each i by the probability gives a polynomial in p with integer coefficients [3]. We used Maple to solve each recurrence (28, 43, 47, 50, 53) algebraically and verified that the polynomials from the recurrences were identical with the polynomials generated from the corresponding node counts.

We verified that the two analyses of the Probe Order Backtracking Algorithm, eqs. (28, 47) and eqs. (50, 53), predicted the same values in two ways. First, we used Maple to solve each recurrence algebraically for $1 \le t \le 6$ and $1 \le v \le 6$ and verified that the formulas were identical. Second, we used each recurrence to compute contours for $v = 50$ and $t \le 179$. The locations of the contours were identical to within the precision to which they were computed. The worst p performance matched to an accuracy of 9 digits.◇

8 Bounds

Simple upper bounds on the running time for Probe Order Backtracking are now computed. The approach is to eliminate indices from the recurrence until one has a simple algebraic equation. To eliminate an index, we assume that the unknown function (T) has a particular dependence on the index being eliminated times a new unknown function of the remaining indices. By plugging the assumed form into the initial recurrence (and performing one or two summations), we obtain a bounding recurrence for the new function.

To simplify the algebra, we now drop the term that starts with δ_{k0} from the definition of H [in eq. (49)] and drop the first negative term from the definition of Z [in eq. (48)]. These changes lead to a new $T(v, m, n)$, which is an upper bound on the running time of the algorithm. They have no significant effect on the computed running time when v is large. Dropping them now saves a lot of ink.

It is convenient to first shift the recurrence by using $T'(v, m, n) = T(v, m, n) - 1$. From eq. (47) we obtain

$$
T'(v,m,n) = \sum_{1\le i\le v} a(v,i)\left(Z(v,i,m,n) + \sum_j\sum_k H(v,i,j,k,m,n)[T'(v-i,j,k)+1]\right)
$$

$$= \sum_{1\le i\le v} a(v,i)\Bigg[Z(v,i,m,n) + \left(\frac{1-(1-p)^{v-i+1}}{1-(1-p)^v}\right)^{m-1}\left(\frac{(1+p)^v-1-p(1+p)^{v-i}}{(1+p)^v-1}\right)^n + \sum_j\sum_k H(v,i,j,k,m,n)T'(v-i,j,k)\Bigg], \tag{56}$$

which can be written as

$$T'(v,m,n) = \sum_{1\le i\le v} a(v,i)\Bigg(Z'(v,i,m,n) + \sum_j\sum_k H(v,i,j,k,m,n)T'(v-i,j,k)\Bigg), \tag{57}$$

where

$$Z'(v,i,m,n) = 2\left(\frac{1-(1-p)^{v-i+1}}{1-(1-p)^v}\right)^{m-1}. \tag{58}$$

The boundary conditions for the shifted recurrence are

$$T'(v,0,n) = T'(v,m,0) = 0, \tag{59}$$

and the average number of nodes is

$$1+\sum_{m,n}\binom{t}{m,n,t-m-n}(1-p)^{(n+m)v}[1-(1-p)^v]^m[(1+p)^v-1]^n[1-(1-p)^v(1+p)^v]^{t-m-n}T'(v,m,n). \tag{60}$$

Details: Summing H over k gives

$$\begin{aligned}\sum_k H(v,i,j,k,m,n) &= \sum_k \left(\frac{(1+p)^{v-i}-1}{(1+p)^v-1}\right)^k \sum_l \binom{m-1}{l}\binom{n}{l+j-m+1}\binom{n+m-l-j-1}{k} \\ &\quad\times\left(\frac{p}{1-(1-p)^v}\right)^l\left(\frac{1-(1-p)^{v-i+1}-p}{1-(1-p)^v}\right)^{m-l-1} \\ &\quad\times\left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^v-1}\right)^{l+j-m+1}\left(\frac{(1+p)^v-(1+p)^{v-i}-p}{(1+p)^v-1}\right)^{n+m-l-j-k-1} \\ &= \sum_l \binom{m-1}{l}\binom{n}{l+j-m+1}\left(\frac{p}{1-(1-p)^v}\right)^l\left(\frac{1-(1-p)^{v-i+1}-p}{1-(1-p)^v}\right)^{m-l-1} \\ &\quad\times\left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^v-1}\right)^{l+j-m+1}\left(\frac{(1+p)^v-1-p}{(1+p)^v-1}\right)^{n+m-l-j-1}.\end{aligned}$$

Summing H over k and j gives

$$\begin{aligned}&\sum_{j,k} H(v,i,j,k,m,n) \\ &= \sum_j\sum_l \binom{m-1}{l}\binom{n}{l+j-m+1}\left(\frac{p}{1-(1-p)^v}\right)^l\left(\frac{1-(1-p)^{v-i+1}-p}{1-(1-p)^v}\right)^{m-l-1} \\ &\quad\times\left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^v-1}\right)^{l+j-m+1}\left(\frac{(1+p)^v-1-p}{(1+p)^v-1}\right)^{n+m-l-j-1} \\ &= \sum_l \binom{m-1}{l}\left(\frac{p}{1-(1-p)^v}\right)^l\left(\frac{1-(1-p)^{v-i+1}-p}{1-(1-p)^v}\right)^{m-l-1} \\ &\quad\times\left(\frac{(1+p)^v-1-p(1-p)^{v-i}}{(1+p)^v-1}\right)^n \\ &= \left(\frac{1-(1-p)^{v-i+1}}{1-(1-p)^v}\right)^{m-1}\left(\frac{(1+p)^v-1-p(1-p)^{v-i}}{(1+p)^v-1}\right)^n.\end{aligned}$$

$$Z'(v,i,m,n) = \left(\frac{1-(1-p)^{v-i+1}}{1-(1-p)^v}\right)^{m-1}\left[2-\left(\frac{(1+p)^v-1-p(1-p)^{v-i}}{(1+p)^v-1}\right)^n\right] + \left(\frac{1-(1-p)^{v-i+1}}{1-(1-p)^v}\right)^{m-1}\left(\frac{(1+p)^v-1-p(1+p)^{v-i}}{(1+p)^v-1}\right)^n,$$

which simplifies to eq. (58). $\diamond$

8.1 Two Index Recurrence

For any $x(v)$ define $T(v,n)$ so that $T'(v,m,n) \le x(v)^m T(v,n)$ for all m. (We could include n dependence in x, but that does not appear to be useful.) Drop the δ_{k0} term from the definition of H (eq. 49) and rearrange the binomials.

Details: To help sum in the j direction, rearrange the binomials in the definition of H as

$$H(v,i,j,k,m,n) = \binom{n}{k}\left(\frac{(1+p)^{v-i}-1}{(1+p)^v-1}\right)^k \sum_l \binom{m-1}{l}\binom{n-k}{l+j-m+1}\left(\frac{p}{1-(1-p)^v}\right)^l \times \left(\frac{1-(1-p)^{v-i+1}-p}{1-(1-p)^v}\right)^{m-l-1}\left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^v-1}\right)^{l+j-m+1} \times \left(\frac{(1+p)^v-(1+p)^{v-i}-p}{(1+p)^v-1}\right)^{n+m-l-j-k-1}.$$

$\diamond$

Combine this H with the definition of Z' (eq. 58). Use the definition of $T(v,n)$ and sum over j to obtain

$$T(v,n) = \frac{1}{x(v)}\max_m\left\{\sum_{1\le i\le v} a(v,i)\left[2\left(\frac{1-(1-p)^{v-i+1}}{x(v)[1-(1-p)^v]}\right)^{m-1} + \left(\frac{x(v-i)[1-(1-p)^{v-i+1}]+[1-x(v-i)]p}{x(v)[1-(1-p)^v]}\right)^{m-1} \times \sum_k \binom{n}{k}\left(\frac{(1+p)^{v-i}-1}{(1+p)^v-1}\right)^k \times \left(\frac{(1+p)^v-(1+p)^{v-i}-[1-x(v-i)]p-x(v-i)p(1-p)^{v-i}}{(1+p)^v-1}\right)^{n-k} T(v-i,k)\right]\right\}. \tag{61}$$

$$T(v,0) = 0, \quad T(v,n) \ge 0. \tag{62}$$

Details: Summing $x^j H$ over j gives

$$
\begin{aligned}
&\sum_j x^j H(v,i,j,k,m,n) \\
&\quad = \sum_j x^j \left[\binom{n}{k} \left(\frac{(1+p)^{v-i}-1}{(1+p)^v-1} \right)^k \sum_l \binom{m-1}{l} \binom{n-k}{l+j-m+1} \left(\frac{p}{1-(1-p)^v} \right)^l \right. \\
&\qquad \times \left(\frac{1-(1-p)^{v-i+1}-p}{1-(1-p)^v} \right)^{m-l-1} \left(\frac{p[1-(1-p)^{v-i}]}{(1+p)^v-1} \right)^{l+j-m+1} \\
&\qquad \left. \times \left(\frac{(1+p)^v-(1+p)^{v-i}-p}{(1+p)^v-1} \right)^{n+m-l-j-k-1} \right] \\
&\quad = \binom{n}{k} \left(\frac{(1+p)^{v-i}-1}{(1+p)^v-1} \right)^k \left(\frac{(1+p)^v-(1+p)^{v-i}-(1-x)p-xp(1-p)^{v-i}}{(1+p)^v-1} \right)^{n-k} \\
&\qquad \times \sum_l \binom{m-1}{l} \left(\frac{p}{1-(1-p)^v} \right)^l x^{m-l-1} \left(\frac{1-(1-p)^{v-i+1}-p}{1-(1-p)^v} \right)^{m-l-1} \\
&\quad = \left(\frac{x[1-(1-p)^{v-i+1}]+(1-x)p}{1-(1-p)^v} \right)^{m-1} \\
&\qquad \times \binom{n}{k} \left(\frac{(1+p)^{v-i}-1}{(1+p)^v-1} \right)^k \left(\frac{(1+p)^v-(1+p)^{v-i}-(1-x)p-xp(1-p)^{v-i}}{(1+p)^v-1} \right)^{n-k}.
\end{aligned}
$$

Eq. (57) implies that we need

$$
x(v)^m T(v,n) \geq \sum_{1 \leq i \leq v} a(v,i) \left(Z(v,i,m,n) + \sum_j \sum_k x(v-i)^j H(v,i,j,k,m,n) T(v-i,k) \right),
$$

$$
T(v,0) = 0, \quad T(v,n) \geq 0,
$$

where the bounds must hold for all m of interest. Thus,

$$
T(v,n) = \max_m \left\{ \frac{1}{x(v)^m} \left[\sum_{1 \leq i \leq v} a(v,i) \left(Z(v,i,m,n) + \sum_j \sum_k x(v-i)^j H(v,i,j,k,m,n) T(v-i,k) \right) \right] \right\},
$$

$$
T(v,0) = 0.
$$

Using the sum of $x^j H$ and the definition of Z, we obtain eq. (61).$\diamond$

So that this recurrence will be favorable, we wish to avoid raising quantities that are above 1 to the m power. Thus, we require

$$
[1-(1-p)^v] x(v) \geq 1-(1-p)^{v-i+1}, \tag{63}
$$

and

$$
[1-(1-p)^v] x(v) \geq [1-(1-p)^{v-i+1}] x(v-i) + [1-x(v-i)]p. \tag{64}
$$

So long as $x(v)$ is above 1, then any increasing function of v can be chosen for $[1-(1-p)^v]x(v)$.

If $x(v)$ obeys the bounds (63, 64), then we may let

$$
\begin{aligned}
T(v,n) = \frac{1}{x(v)} \left[\sum_{1 \leq i \leq v} a(v,i) \left(2 + \sum_k \binom{n}{k} \left(\frac{(1+p)^{v-i}-1}{(1+p)^v-1} \right)^k \right. \right. \\
\left. \left. \times \left(\frac{(1+p)^v-(1+p)^{v-i}-[1-x(v-i)]p-x(v-i)p(1-p)^{v-i}}{(1+p)^v-1} \right)^{n-k} T(v-i,k) \right) \right].
\end{aligned} \tag{65}
$$

Eq. (60) implies the average number of nodes is bounded by

$$1+\sum_{m,n}\binom{t}{m,\,n,\,t-m-n}(1-p)^{(n+m)v}x(v)^m[1-(1-p)^v]^m \\ \times[(1+p)^v-1]^n[1-(1-p)^v(1+p)^v]^{t-m-n}T(v,n) \\ =1+\sum_n\binom{t}{n}(1-p)^{nv}[(1+p)^v-1]^n\{1-(1-p)^v(1+p)^v+x(v)(1-p)^v[1-(1-p)^v]\}^{t-n}T(v,n). \tag{66}$$

Eq. (66) gives a good bound when $x(v)$ is set to the average length of an all positive clause, eq. (17). Figure 5 shows the bounds that result from this value of x.

Details:

Figure 5a shows the bound when

$$x(v)=\frac{a(p,t,v)pv}{1-(1-p)^v}$$

and $a(p,t,v)$ has the value computed at the end of Section 8.3.$\diamond$

Note the division by $x(v)$ in eq. (65). This is critical to obtaining an analytical understanding of why Probe Order Backtracking is fast. We are free to set $x(v)$ large enough to cancel out the effect of summing over i (which is where the growth in $T(v,n)$ comes from) so long as the factor in eq. (66) which is raised to the $t-n$ power is not above 1. This division by $x(v)$ is related to the fact that selecting an all positive clause results in a reduction of one in the number of all positive clauses (the setting of variables can augment or counteract this reduction). In Backtracking with Probing we do not have this tendency to reduce the number of all positive clauses by 1, and thus that algorithm is often much slower.

8.2 One Index Recurrence

For any $x(v)$ and $y(v)$ define $T(v)$ so that $T'(v,m,n)\le x(v)^m y(v)^n T(v)$ for all m and n. Then a suitable $T(v)$ is any function at least as large as the solution to

$$T(v)=\frac{1}{x(v)}\max_{m,n}\Bigg\{\sum_{1\le i\le v}a(v,i) \\ \times\Bigg[\frac{2}{y(v)^n}\left(\frac{1-(1-p)^{v-i+1}}{x(v)[1-(1-p)^v]}\right)^{m-1} \\ +\left(\frac{x(v-i)[1-(1-p)^{v-i+1}]+[1-x(v-i)]p}{x(v)[1-(1-p)^v]}\right)^{m-1} \\ \times\left(\frac{(1+p)^v+[y(v-i)-1](1+p)^{v-i}-y(v-i)-[1-x(v-i)]p-x(v-i)p(1-p)^{v-i}}{y(v)[(1+p)^v-1]}\right)^n \\ \times T(v-i)\Bigg]\Bigg\}. \tag{67}$$

Details: Summing x^jy^kH over j and k gives

$$\sum_{j,k}x^jy^kH(v,i,j,k,m,n) \\ =\sum_k y^k\left(\frac{x[1-(1-p)^{v-i+1}]+(1-x)p}{1-(1-p)^v}\right)^{m-1} \\ \times\binom{n}{k}\left(\frac{(1+p)^{v-i}-1}{(1+p)^v-1}\right)^k\left(\frac{(1+p)^v-(1+p)^{v-i}-(1-x)p-xp(1-p)^{v-i}}{(1+p)^v-1}\right)^{n-k} \\ =\left(\frac{x[1-(1-p)^{v-i+1}]+(1-x)p}{1-(1-p)^v}\right)^{m-1} \\ \times\left(\frac{(1+p)^v+(y-1)(1+p)^{v-i}-y-(1-x)p-xp(1-p)^{v-i}}{(1+p)^v-1}\right)^n.$$

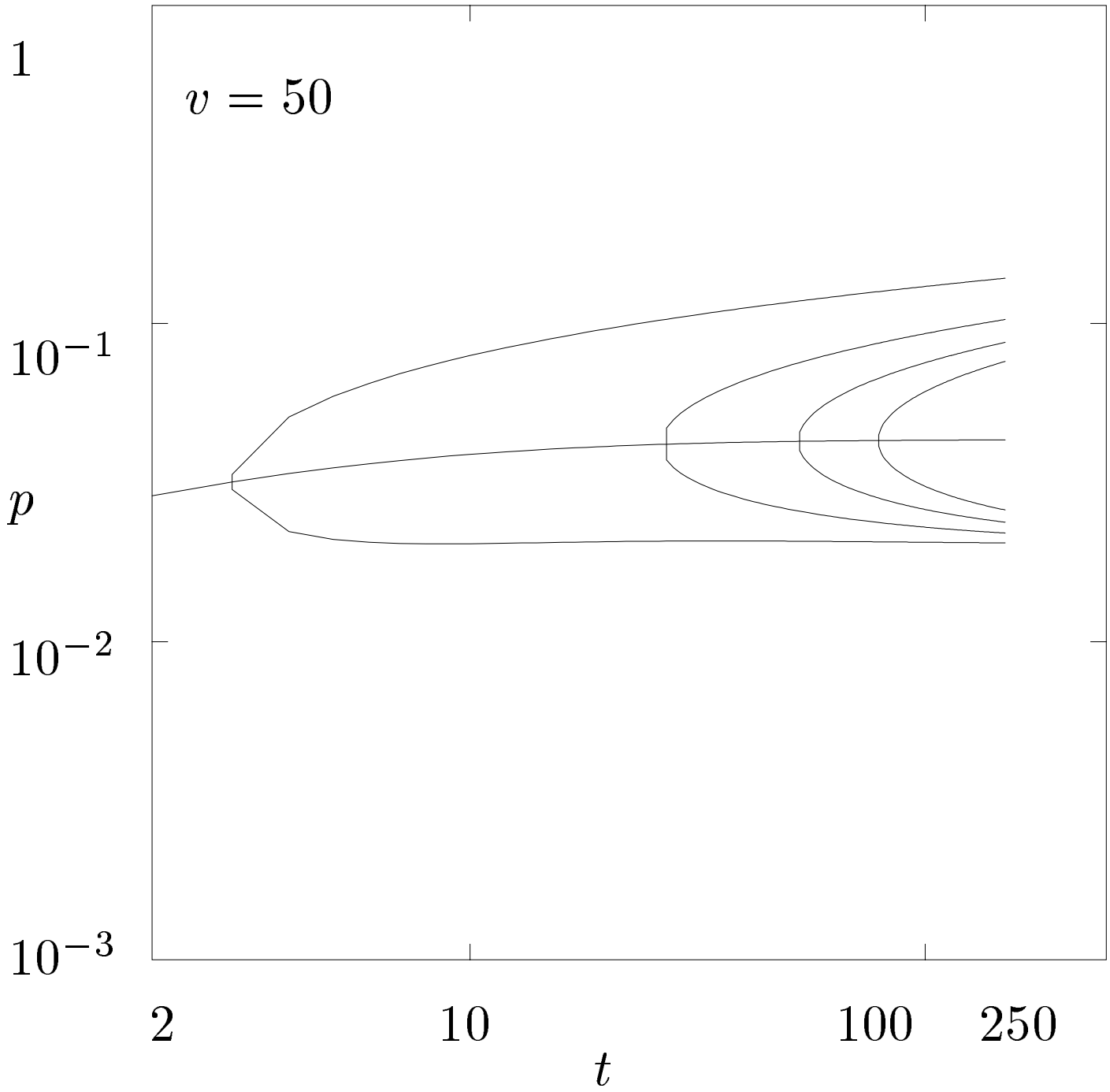

Fig. 5. Two index upper limit.

Using this sum in eq. (61) gives eq. (67). $\diamond$

Again, we wish to avoid raising quantities above 1 to high powers. Thus, we still have bounds (63, 64) for $x(v)$. In addition we have

$$y(v) \geq 1, \tag{68}$$

and

$$[y(v)-1][(1+p)^v - 1] - [y(v-i)-1][(1+p)^{v-i} - 1] \geq p\{x(v-i)[1-(1-p)^{v-i}] - 1\}. \tag{69}$$

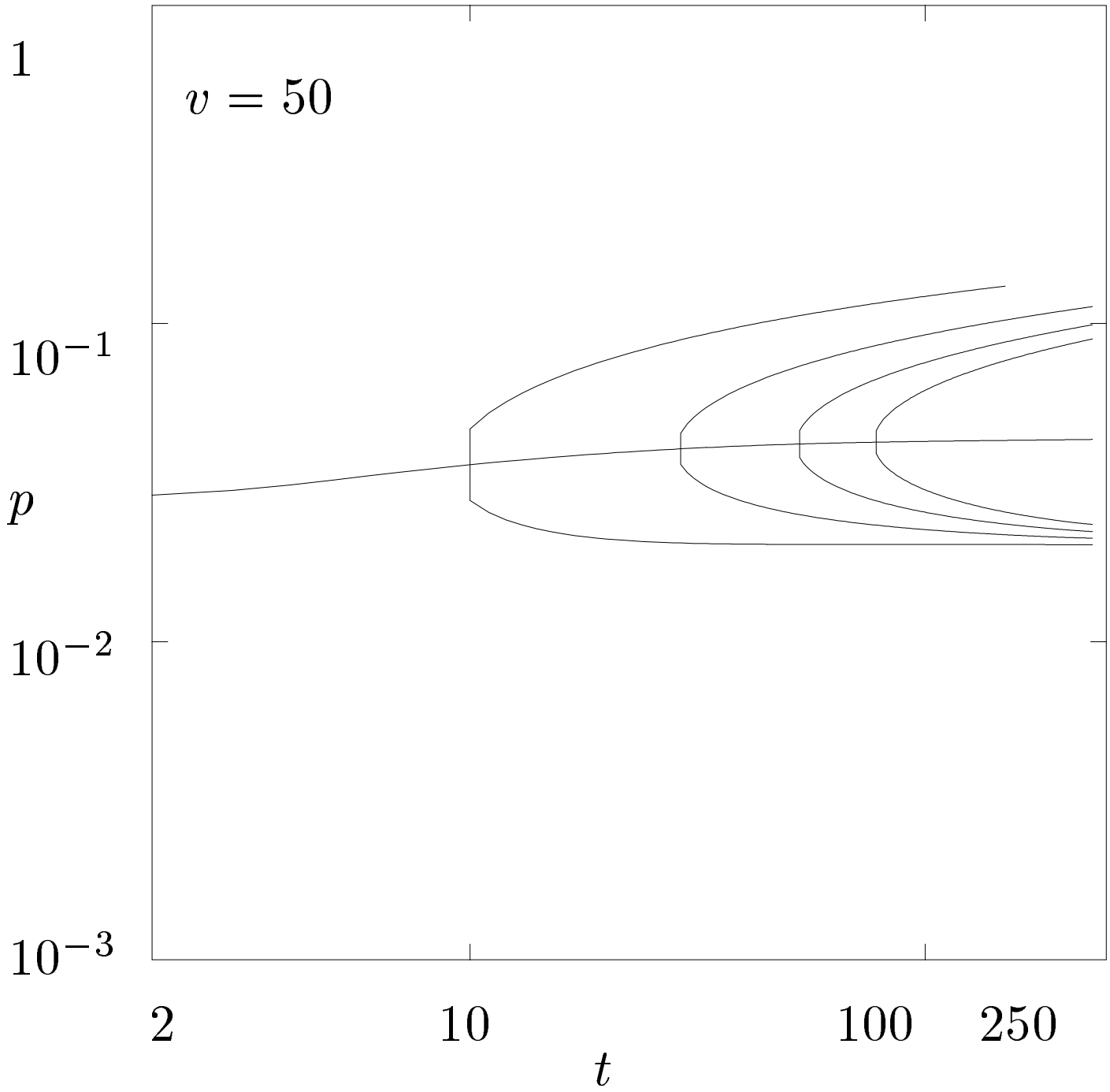

Fig. 5a. Two index upper limit, improved x.

These bounds for y are satisfied by

$$[y(v)-1][(1+p)^v-1] = p \sum_{1 \le j \le v-1} \max\{0, \{x(j)[1-(1-p)^j]-1\}\}. \tag{70}$$

If $x(v)$ and $y(v)$ obey the bounds, we have

$$T(v) = \frac{1}{x(v)} \sum_{1 \le i \le v} a(v,i)[2+T(v-i)]. \tag{71}$$

Eq. (66) implies the average number of nodes is bounded by

$$1+\sum_n \binom{t}{n} y(v)^n (1-p)^{nv} [(1+p)^v - 1]^n \{1-(1-p)^v(1+p)^v + x(v)(1-p)^v[1-(1-p)^v]\}^{t-n} T(v)$$
$$= 1 + \{1-(1-p)^v(1+p)^v + y(v)(1-p)^v[(1+p)^v-1] + x(v)(1-p)^v[1-(1-p)^v]\}^t T(v). \quad (72)$$

If the value of $y(v)$ is set by eq. (70), then the number of nodes is bounded by

$$1+\left[1+(1-p)^v\left(x(v)[1-(1-p)^v]-1+p\sum_{1\le j\le v-1}\max\{0,(x(j)[1-(1-p)^j]-1)\}\right)\right]^t T(v). \quad (73)$$

Eq. (73) gives a good bound when $x(v)$ is set to the average length of an all positive clause, eq. (17). Figure 6 shows the bounds that result from this value of x.

Details:

Figure 6a shows the bound when x is given an improved value that is discussed in Section 8.3. $\diamond$

If one ignores the requirement that $y(v)$ satisfy bounds (68, 69) and just sets $x(v)$ to the average clause size and $y(v) = 1$, one obtains a result that is essentially the same as that given by the heuristic analysis, eq. (30).

8.3 Zero Indices

Eq. (71) has only one index, but it is still rather complex due to the summation on the right side. Therefore, we will again eliminate an index from the recurrence.

Assume $T(v)$ is no more than T for $v < v_*$. (This assumption does not lead to much error when T is small; if one wishes a good approximation when T is large, one should consider $T(v) \le Tz^v$ and select the best value for z.) We obtain

$$T(v_*) \le \frac{2+T}{x(v)} \sum_{1\le i\le v} a(v,i). \quad (74)$$

A good choice for $x(v)$ is one that cancels the effect of the summation. For

$$x(v) = \frac{pv}{1-(1-p)^v}, \quad (75)$$

we have

$$\sum_{1\le j\le v-1} \max\{x(j)[1-(1-p)^j]-1, 0\} = \sum_{1/p\le j\le v-1} (pj-1)$$
$$\le \frac{pv(v-1)}{2} - \frac{(1/p-1)}{2} - \left(v-\frac{1}{p}\right), \quad (76)$$

(the less than or equal comes from the fact that $1/p$ may be a noninteger). Thus, eqs. (73, 76) imply that the number of nodes is bounded by

$$T(v_*) = 2+T, \quad (77)$$

which is satisfied for all v if we take

$$T(v) = 2v. \quad (78)$$

Details: From the definition of $a(v,i)$, eq. (44),

$$\sum_{i\ge 1} a(v,i) = \sum_{i\ge 1}\sum_{j\ge i}\binom{v}{j}\frac{p^j(1-p)^{v-j}}{1-(1-p)^v} = \sum_{j\ge 1}\sum_{1\le i\le j}\binom{v}{j}\frac{p^j(1-p)^{v-j}}{1-(1-p)^v}$$
$$= v\sum_{j\ge 1}\binom{v-1}{j-1}\frac{p^j(1-p)^{v-j}}{1-(1-p)^v} = \frac{pv}{1-(1-p)^v}.$$

$\diamond$

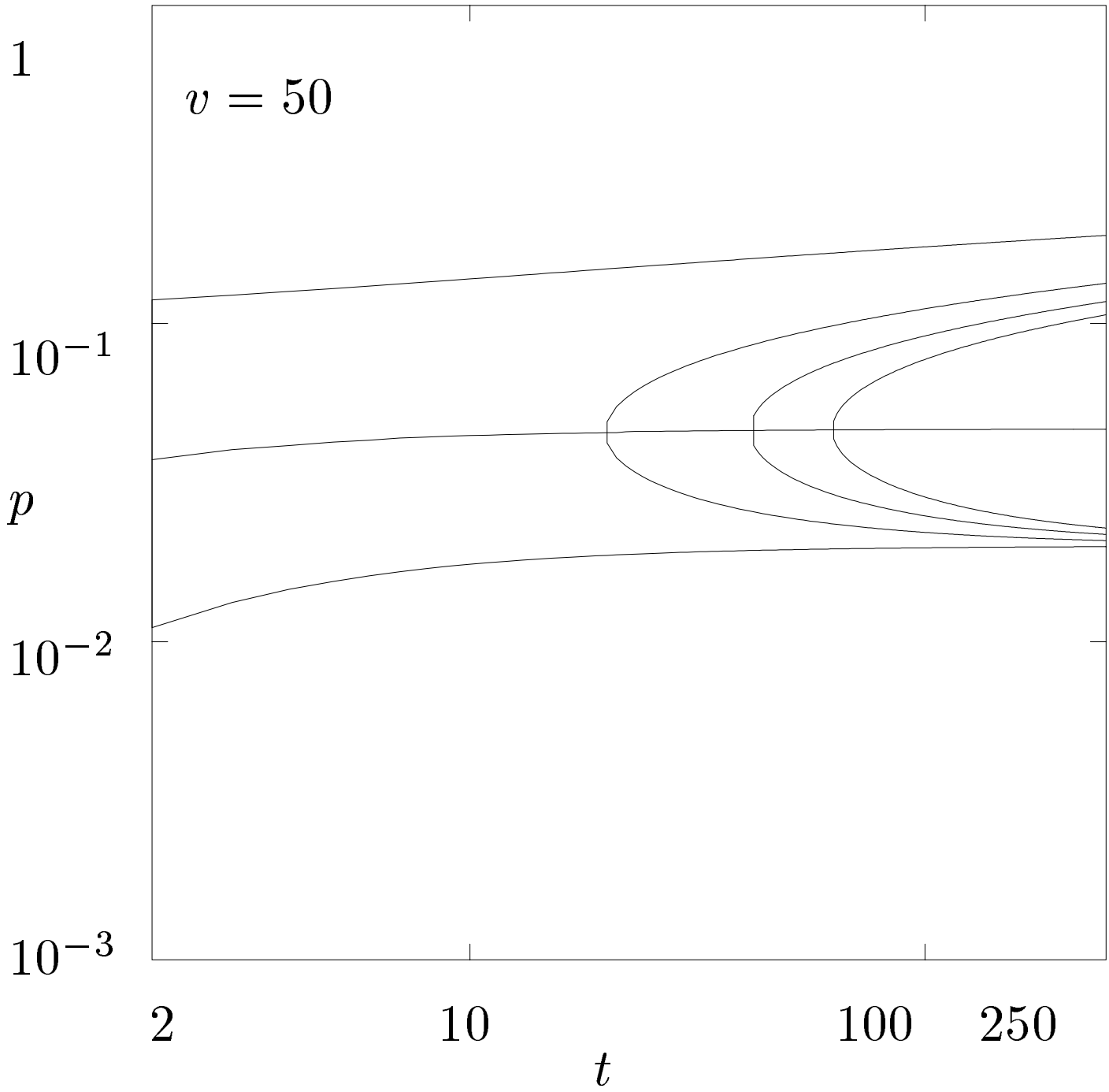

Fig. 6. One index upper limit.

Thus, the number of nodes is bounded by

$$1 + 2v \left[1 + (1-p)^v \left(\frac{p^2 v(v-1)}{2} - \frac{1-p}{2} \right) \right]^t . \tag{79}$$

Figure 7 shows the bounds that result from eq. (79).
If we take

$$x(v) = \frac{apv}{1-(1-p)^v} \tag{80}$$

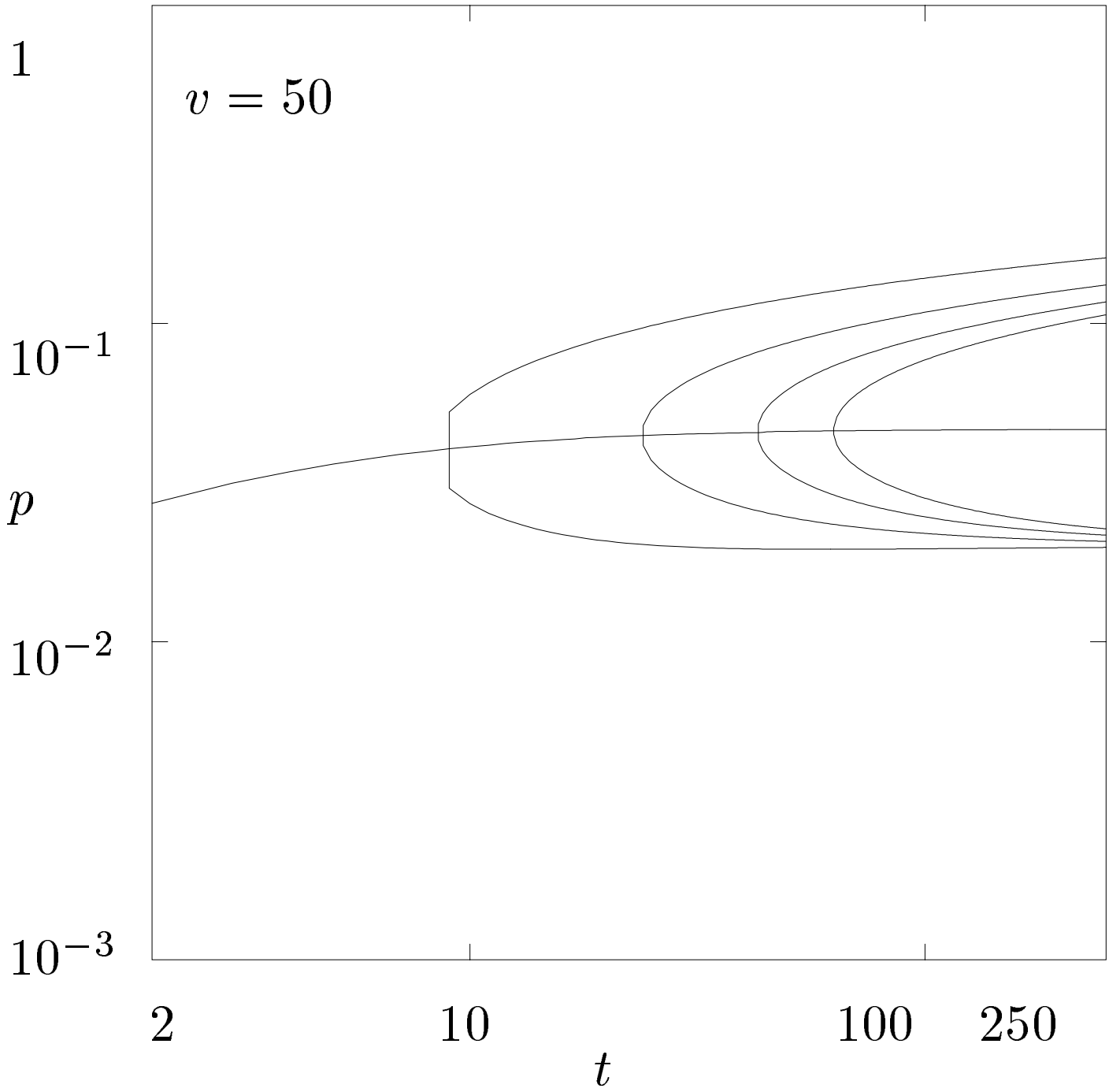

Fig. 6a. One index upper limit, improved x.

with $a > 1$, we have

$$T(v_*) = \frac{2+T}{a}, \tag{81}$$

which is satisfied for all v if we take

$$T(v) = \frac{2}{a-1}. \tag{82}$$

From eq. (80) we have

$$\sum_{1 \le j \le v-1} \max\{x(j)[1-(1-p)^j]-1, 0\} = \sum_{1/(ap) \le j \le v-1} (apj-1)$$

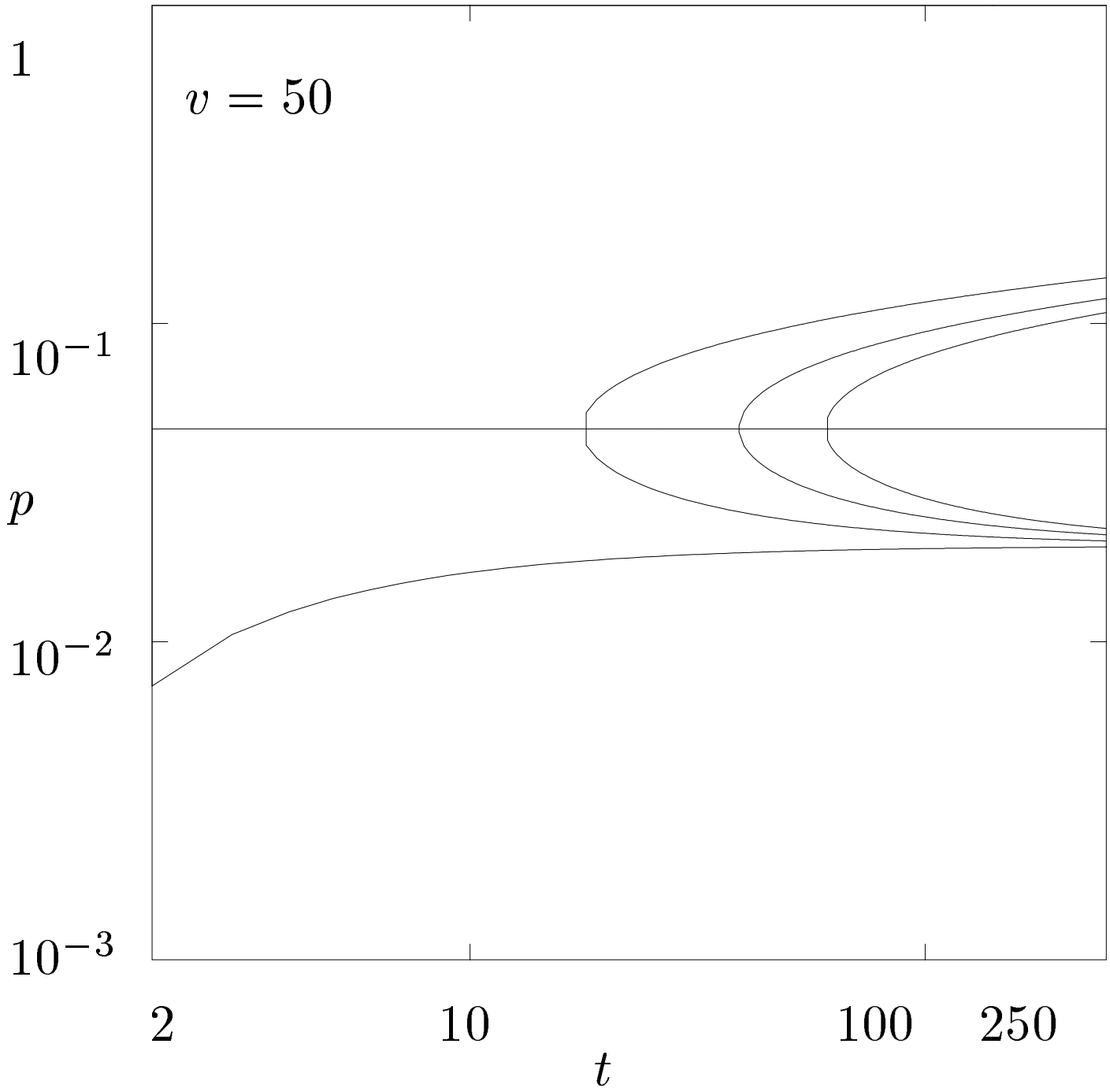

Fig. 7. Zero index upper limit.

$$\leq \frac{apv(v-1)}{2} - \frac{(1/(ap)-1)}{2} - \left(v - \frac{1}{ap}\right). \tag{83}$$

Eq. (73) implies the average number of nodes is bounded by

$$1 + \frac{2}{a-1}\left[1 + (1-p)^v\left(\frac{1/a+p}{2} - 1 + (a-1)pv + \frac{ap^2v(v-1)}{2}\right)\right]^t. \tag{84}$$

The derivative of this equation is cubic in a, but if we replace the $1/a$ term with one, then the derivative is

a linear function. This suggests using

$$a = \frac{2 + (1-p)^v [-1 + p + 2(t-1)pv + p^2 tv(v-1)]}{p(t-1)v(1-p)^v [p(v-1)+2]} \tag{85}$$

in eq. (84).

Details: A Maple calculation shows that the derivative of (84) gives an equation that is cubic in a. The a that minimizes

$$1 + \frac{2}{a-1}\left[1 + (1-p)^v \left(\frac{1+p}{2} - 1 + (a-1)pv + \frac{ap^2 v(v-1)}{2}\right)\right]^t$$

is given by eq. (85).$\diamond$

Figure 8 shows the bounds that result from eqs. (84, 85).

9 Asymptotics

Since eqs. (84, 85) are more complex than eq. (79) the asymptotic analysis is based on eq. (79). We require that the bound on the number of nodes be no more than v^n. That is,

$$v^n \geq 1 + 2v\left[1 + (1-p)^v \left(\frac{p^2 v(v-1)}{2} - \frac{1-p}{2}\right)\right]^t, \tag{86}$$

or

$$\frac{v^{n-1}}{2}\left(1 - \frac{1}{v^n}\right) \geq \left[1 + (1-p)^v \left(\frac{p^2 v(v-1)}{2} - \frac{1-p}{2}\right)\right]^t. \tag{87}$$

9.1 Small t

Solving for t in bound (87) gives

$$t \leq \frac{(n-1)\ln v - \ln 2 - \Theta(1/v^n)}{\ln\{1 + [p^2 v(v-1) - 1 + p](1-p)^v/2\}}. \tag{88}$$

Details: From eq. (87)

$$\ln\left[\frac{v^{n-1}}{2}\left(1 - \frac{1}{v^n}\right)\right] \geq t \ln\left[1 + (1-p)^v \left(\frac{p^2 v(v-1)}{2} - \frac{1-p}{2}\right)\right].$$

Also

$$\ln\left[\frac{v^{n-1}}{2}\left(1 - \frac{1}{v^n}\right)\right] = (n-1)\ln v - \ln 2 - \Theta\left(\frac{1}{v^n}\right).$$

$\diamond$

For $n > 1$ and $1 < a \leq pv \leq b < \infty$, (88) can be simplified to

$$t \leq \frac{(n-1)\ln v - \ln 2}{\ln\{1 + [(pv)^2 - 1]e^{-pv}/2\}}, \tag{89}$$

which is bound (4).

Details: Since $pv \leq b$

$$(1-p)^v = e^{v\ln(1-p)} = e^{-pv}e^{-v\Theta(p^2)} = e^{-pv}[1 - \Theta(p)],$$

$$\begin{aligned}\ln\left(1 + (1-p)^v \frac{p^2 v(v-1) - 1 + p}{2}\right) &= \ln\left\{1 + e^{-pv}\frac{(pv)^2 - 1}{2}\left[1 - \Theta(p) - \Theta\left(\frac{1}{v}\right)\right]\right\} \\ &= \ln\left\{1 + e^{-pv}\frac{(pv)^2 - 1}{2}\left[1 - \Theta\left(\frac{1}{v}\right)\right]\right\}.\end{aligned}$$

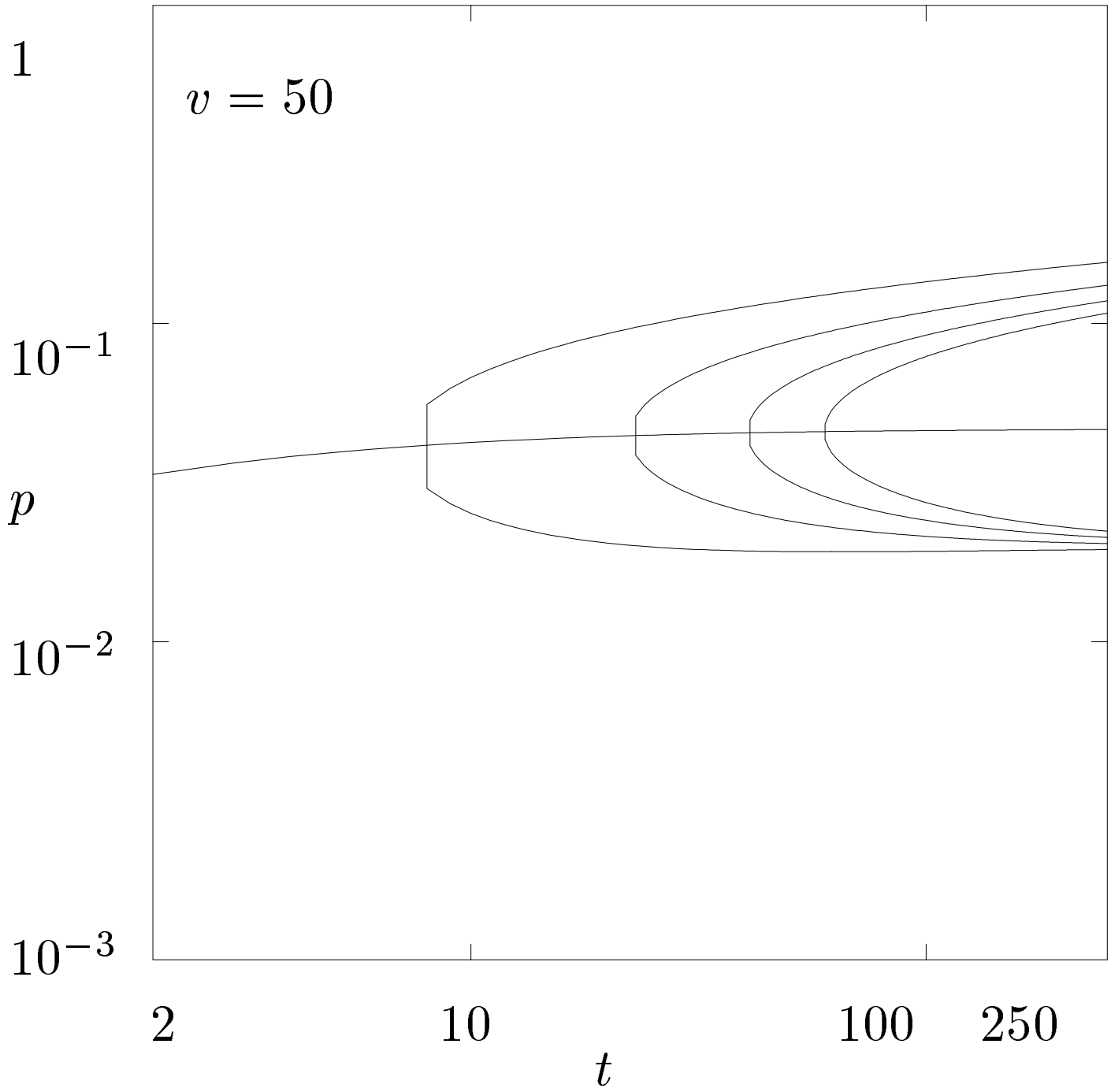

Fig. 8. Improved zero index upper limit.

Now

$$\begin{aligned}\ln\{1 + y[1 + \Theta(x)]\} &= \ln(1 + y)[1 + \Theta(x)/(1 + y)] \\ &= \ln(1 + y) + \Theta(x)/(1 + y).\end{aligned}$$

Thus, when $1 < a \leq y \leq b < \infty$

$$\ln\{1 + y[1 + \Theta(x)]\} = \ln(1 + y) + \Theta(x)$$

and

$$\frac{1}{\ln\{1 + y[1 + \Theta(x)]\}} = \frac{1 - \Theta(x)}{\ln(1 + y)}.$$

Since

$$(1-p)^v \frac{p^2v(v-1)-1+p}{2}$$

is positive and bounded for $1 < a \le pv \le b < \infty$ and large v, we have

$$\frac{(n-1)\ln v - \ln 2 - \Theta(1/v^n)}{\ln\{1+(1-p)^v[p^2v(v-1)-1+p]/2\}} = \frac{(n-1)\ln v - \ln 2 - \Theta(1/v^n) + \Theta(1/v)}{\ln\{1+[(pv)^2-1]e^{-pv}/2\}}.$$

When $n > 1$ and v is large the $\Theta(1/v)$ term is more important than the $\Theta(1/v^n)$ term and since this is an upper bound, positive Θ terms can be dropped.$\diamond$

9.2 Small p

From bound (87) using algebra, $x = e^{\ln x}$, and power series we obtain

$$(1-p)^v[p^2v(v-1)-1+p] \le \frac{2[(n-1)\ln v - \ln 2]}{t}\left[1+\Theta\left(\frac{\ln v}{t}\right)\right]. \tag{90}$$

Details:

$$\begin{aligned}
1+(1-p)^v\left(\frac{p^2v(v-1)}{2}-\frac{1-p}{2}\right) &\le \frac{v^{(n-1)/t}}{2^{1/t}}\left[1-\Theta\left(\frac{1}{v^n t}\right)\right] \\
&\le e^{[(n-1)\ln v - \ln 2 - \Theta(1/v^n)]/t} \\
&\le 1+\frac{(n-1)\ln v - \ln 2}{t}\left[1+\Theta\left(\frac{\ln v}{t}\right)\right].
\end{aligned}$$

Thus,

$$(1-p)^v\left(\frac{p^2v(v-1)}{2}-\frac{1-p}{2}\right) \le \frac{(n-1)\ln v - \ln 2}{t}\left[1+\Theta\left(\frac{\ln v}{t}\right)\right].$$

$\diamond$

When $pv > 1$ and p^2v is bounded, bound (90) can be written as

$$[(pv)^2-1]e^{-pv} \le \frac{2[(n-1)\ln v - \ln 2]}{t}\left[1+\Theta\left(\frac{1}{v}\right)+\Theta(p^2v)+\Theta\left(\frac{\ln v}{t}\right)\right]. \tag{91}$$

Details: Twice the left side of bound (90) is

$$(1-p)^v[p^2v(v-1)-1+p] = (pv-1)(pv+1-p)e^{-pv}e^{-\Theta(p^2v)}.$$

Since $pv > 1$, we have $pv+1-p = (pv+1)[1-\Theta(1/v)]$ and since p^2v is bounded we have $e^{-\Theta(p^2v)} = 1-\Theta(p^2v)$. Thus, the left side of bound (90) is

$$[(pv)^2-1]e^{-pv}\left[1-\Theta\left(\frac{1}{v}\right)-\Theta(p^2v)\right],$$

and we can write bound (90) as bound (91).$\diamond$

When pv is near 1, bound (91) is equivalent to

$$pv \le 1+\frac{e[(n-1)\ln v - \ln 2]}{t} \tag{92}$$

which is equivalent to bound (3).

Details: Bound (91) can be written as

$$p^2v^2 \le 1+2e^{pv}\frac{(n-1)\ln v - \ln 2}{t}\left[1+\Theta\left(\frac{1}{v}\right)+\Theta\left(\frac{\ln v}{t}\right)\right].$$

To find the solution near $pv = 1$, let $pv = 1+y$, giving

$$2y+y^2 \le 2e[1+y+\Theta(y^2)]\frac{(n-1)\ln v - \ln 2}{t}\left[1+\Theta\left(\frac{1}{v}\right)+\Theta\left(\frac{\ln v}{t}\right)\right].$$

Thus,

$$y \le \frac{e[(n-1)\ln v - \ln 2]}{t}\left[1+\Theta\left(\frac{1}{v}\right)+\Theta\left(\frac{\ln v}{t}\right)\right].$$

Since this is an upper bound, we drop positive Θ terms to obtain bound (92). (There is no solution with pv below 1.) $\diamond$

9.3 Large p

In the previous section we found a solution to bound (87) that has pv near 1. For large pv, $(1-p)^v$ decreases much more rapidly than $(pv)^2$ increases. Bound (91) has the form $(x^2-1)e^{-x} \le y$ with small y. The large x solution is

$$x \ge -\ln y + 2\ln(-\ln y) + \Theta\left(\frac{\ln(-\ln y)}{-\ln y}\right). \tag{93}$$

Details: Assume $x = -\ln y + 2\ln(-\ln y) + z\ln(-\ln y)/(-\ln y)$. Plugging into $(x^2-1)e^{-x} \le y$ gives

$$\frac{[-\ln y + 2\ln(-\ln y) + z\ln(-\ln y)/(-\ln y)]^2 - 1}{(-\ln y)^{2+z/(-\ln y)}} y = y.$$

Dividing both sides by y and clearing fractions gives

$$\left(1 + \frac{2\ln(-\ln y)}{-\ln y} + \frac{z\ln(-\ln y)}{(-\ln y)^2}\right)^2 - \frac{1}{(-\ln y)^2} = (-\ln y)^{z/(-\ln y)}.$$

Taking logarithms, expanding the logarithms, and retaining the important terms gives

$$\frac{4\ln(-\ln y)}{-\ln y}\left[1 + \Theta\left(\frac{\ln(-\ln y)}{-\ln y}\right)\right] = \frac{z\ln(-\ln y)}{-\ln y}.$$

In the limit the right side is bigger for $z > 4$ and the left side is bigger for $z \le 4$. Thus,

$$x = -\ln y + 2\ln(-\ln y) + \Theta\left(\frac{\ln(-\ln y)}{-\ln y}\right)$$

is a solution. (Also, there is no solution with larger x.)$\diamond$

When t increases more rapidly than $\ln v$ the solution to (91) is

$$pv \ge \ln t + 2\ln\ln t - \ln\ln v - \ln(n-1) - \ln 2 + \Theta\left(\frac{\ln\ln t}{\ln t}\right). \tag{94}$$

This is bound (1).

Details: For eq. (93) we have

$$-\ln y = \ln t - \ln\ln v - \ln(n-1) - \ln 2 - \Theta\left(\frac{1}{v}\right) - \Theta(p^2 v) - \Theta\left(\frac{\ln v}{t}\right),$$

and

$$\ln(-\ln y) = \ln\ln t - \Theta\left(\frac{\ln\ln v}{\ln t}\right) - \Theta\left(\frac{p^2 v}{\ln t}\right) - \Theta\left(\frac{\ln v}{t\ln t}\right),$$

so, when t increases more rapidly than $\ln v$

$$pv \ge \ln t + 2\ln\ln t - \ln\ln v - \ln(n-1) - \ln 2 - \Theta\left(\frac{\ln\ln v}{\ln t}\right) - \Theta\left(\frac{1}{v}\right) - \Theta(p^2 v) - \Theta\left(\frac{\ln v}{t}\right) + \Theta\left(\frac{\ln\ln t}{\ln t}\right).$$

Since this is a lower bound the negative Θ terms can be dropped.$\diamond$

9.4 Comparison with Simple Backtracking

When t/v is large, the results of the small p analysis are not very good. A better result can be obtained by observing that the average running time for Probe Order Backtracking is no larger than that of Simple Backtracking. (The proof that the average running time of Clause Order Backtracking is no larger than that of Simple Backtracking [3, Theorem 1] also applies to Probe Order Backtracking.)

We require that A.18 from [22], the bound for Simple Backtracking, be no more than v^n. For this bound we use $M(v) = v$ and $\delta = 0$ and let $q = -\ln(1-p)$ to obtain

$$v^n \geq 1 + v \exp\left[2(\ln 2)v + \ln 2 - \frac{\ln 2}{q} \ln\left(1 + \frac{qt}{\ln 2}\right) + t \ln\left(1 - \frac{\ln 2}{\ln 2 + qt}\right)\right]. \tag{95}$$

This can be written as

$$qv - \frac{1}{2}\ln(qv) \leq \frac{1}{2}\ln\left(\frac{t}{v}\right) + \frac{1 - \ln\ln 2}{2} + \frac{q}{2\ln 2}[(n-1)\ln v - \ln 2] - \Theta\left(\frac{q}{v^n}\right) + \Theta\left(\frac{1}{qt}\right). \tag{96}$$

Details: When qt is large, it is useful to write the right side of (95) as

$$\begin{aligned}
&1 + v \exp\left[2(\ln 2)v + \ln 2 - \frac{\ln 2}{q} \ln\left(\frac{qt + \ln 2}{\ln 2}\right) + t \ln\left(\frac{qt}{\ln 2 + qt}\right)\right] \\
&= 1 + v \exp\left[2(\ln 2)v + \ln 2 + \frac{(\ln 2)\ln\ln 2}{q} - \left(\frac{\ln 2}{q} + t\right)\ln(qt + \ln 2) + t\ln(qt)\right] \\
&= 1 + v \exp\left[\left(\frac{1}{q}\right)[2(\ln 2)qv + q\ln 2 + (\ln 2)\ln\ln 2 - (qt + \ln 2)\ln(qt + \ln 2) + qt\ln(qt)]\right] \\
&= 1 + v \exp\left\{\left(\frac{1}{q}\right)\left[2(\ln 2)qv + q\ln 2 + (\ln 2)\ln\ln 2 - (\ln 2)\ln(qt) - (qt + \ln 2)\ln\left(1 + \frac{\ln 2}{qt}\right)\right]\right\}.
\end{aligned}$$

Now

$$\begin{aligned}
(x + \ln 2)\ln\left(1 + \frac{\ln 2}{x}\right) &= (x + \ln 2)\left[\frac{\ln 2}{x} - \frac{1}{2}\left(\frac{\ln 2}{x}\right)^2 + \Theta\left(\frac{1}{x^3}\right)\right] \\
&= \ln 2 - \frac{(\ln 2)^2}{2x} + \Theta\left(\frac{1}{x^2}\right) + \frac{(\ln 2)^2}{x} - \Theta\left(\frac{1}{x^2}\right) \\
&= \ln 2 + \Theta\left(\frac{1}{x}\right),
\end{aligned}$$

so the right side is

$$1 + v \exp\left\{\left(\frac{1}{q}\right)\left[2(\ln 2)qv - \ln 2 + (\ln 2)\ln\ln 2 - (\ln 2)\ln(qt) + q\ln 2 - \Theta\left(\frac{1}{qt}\right)\right]\right\}$$

Using this in bound (95), rearranging, taking logarithms, and multiplying by q gives

$$(n-1)q\ln v \geq 2(\ln 2)qv - \ln 2 + (\ln 2)\ln\ln 2 - (\ln 2)\ln(qt) + q\ln 2 - \Theta\left(\frac{1}{qt}\right) + \Theta\left(\frac{q}{v^n}\right).$$

Writing qt as $(qv)(t/v)$, separating qv terms, and dividing by $2\ln 2$ gives (96).$\diamond$

When $t/v > e$ the solution to bound (96) is

$$qv \leq \frac{1}{2}\ln\left(\frac{t}{v}\right) + \frac{1}{2}\ln\ln\left(\frac{t}{v}\right) + \frac{1 - \ln 2 - \ln\ln 2}{2} + \frac{q}{2\ln 2}[(n-1)\ln v - \ln 2] + \Theta\left(\frac{\ln\ln(t/v)}{\ln(t/v)}\right) - \Theta\left(\frac{q}{v^n}\right). \tag{97}$$

Details: Consider the test solution

$$qv = \frac{1}{2}\ln\left(\frac{t}{v}\right) + \frac{1}{2}\ln\ln\left(\frac{t}{v}\right) + \frac{1 - \ln 2 - \ln\ln 2}{2} + \frac{q}{2\ln 2}[(n-1)\ln v - \ln 2] + a\frac{\ln\ln(t/v)}{\ln(t/v)} - \Theta\left(\frac{q}{v^n}\right).$$

Assuming $q \ln v$ is small

$$\begin{aligned}\ln(qv) &= \ln\ln\left(\frac{t}{v}\right) - \ln 2 + \ln\left[1 + \frac{\ln\ln(t/v)}{\ln(t/v)} + \Theta\left(\frac{1}{\ln(t/v)}\right)\right] \\ &= \ln\ln\left(\frac{t}{v}\right) - \ln 2 + \frac{\ln\ln(t/v)}{\ln(t/v)} + \Theta\left(\frac{1}{\ln(t/v)}\right).\end{aligned}$$

Plugging the test solution into eq. (96) and simplifying gives

$$a\frac{\ln\ln(t/v)}{\ln(t/v)} - \frac{\ln\ln(t/v)}{2\ln(t/v)} \le \Theta\left(\frac{1}{qt}\right) + \Theta\left(\frac{1}{\ln(t/v)}\right).$$

The $\Theta(q/v^n)$ terms cancel since they have the same implied constant. When $t/v > e$, $\ln\ln(t/v) > 0$. Thus, when $t/v > e$, the test solution, or anything smaller, works in the limit for any $a < 1/2$ giving bound (97).$\diamond$

Replacing q with its value in terms of p and solving for p in bound (97) gives

$$\begin{aligned}p \le &\left[\frac{\ln(t/v) + \ln\ln(t/v) + 1 - \ln 2 - \ln\ln 2}{2v}\right] \\ &\times \left[1 + \frac{2(n-1)\ln v - (\ln 2)[\ln(t/v) + \ln\ln(t/v) + 3 - \ln 2 - \ln\ln 2]}{4v\ln 2}\right. \\ &\left. - \Theta\left(\frac{(\ln v)\ln(t/v)}{v^2}\right) - \Theta\left(\frac{\ln\ln(t/v)}{v\ln(t/v)}\right)\right] \qquad (98)\end{aligned}$$

which is bound (2). For large v this is an improvement over the small p analysis when $t = \beta v$ and $\beta > 3.22136$. Note that $\beta > e$.

Details: Solving bound (97) for q gives

$$q \le \left[\frac{\ln(t/v) + \ln\ln(t/v) + 1 - \ln 2 - \ln\ln 2}{2v} + \Theta\left(\frac{\ln\ln(t/v)}{v\ln(t/v)}\right)\right]\left[1 + \frac{(n-1)\ln v - \ln 2}{2v\ln 2} + \Theta\left(\frac{(\ln v)^2}{v^2}\right)\right].$$

From the definition of q

$$q = -\ln(1-p) = p + \frac{p^2}{2} + \frac{p^3}{3} + \Theta(p^4).$$

For small y, the solution, with p near y, to

$$p + \frac{p^2}{2} + \frac{p^3}{3} + \Theta(p^4) = y$$

is

$$p = y\left[1 - \frac{y}{2} + \Theta(y^2)\right].$$

Hence, solving for p in bound (97) gives

$$\begin{aligned}p \le &\left[\frac{\ln(t/v) + \ln\ln(t/v) + 1 - \ln 2 - \ln\ln 2}{2v} + \Theta\left(\frac{\ln\ln(t/v)}{v\ln(t/v)}\right)\right]\left[1 + \frac{(n-1)\ln v - \ln 2}{2v\ln 2} + \Theta\left(\frac{(\ln v)^2}{v^2}\right)\right] \\ &\times \left\{1 - \frac{\ln(t/v) + \ln\ln(t/v) + 1 - \ln 2 - \ln\ln 2}{4v} - \Theta\left(\frac{(\ln v)\ln(t/v)}{v^2}\right)\right. \\ &\left. - \Theta\left(\frac{\ln\ln(t/v)}{v\ln(t/v)}\right) + \Theta\left(\left[\frac{\ln(t/v)}{v}\right]^2\right)\right\}.\end{aligned}$$

Multiplying the last two terms gives

$$\begin{aligned}p \le &\left[\frac{\ln(t/v) + \ln\ln(t/v) + 1 - \ln 2 - \ln\ln 2}{2v} + \Theta\left(\frac{\ln\ln(t/v)}{v\ln(t/v)}\right)\right] \\ &\times \left\{1 + \frac{2(n-1)\ln v - 2\ln 2 - (\ln 2)[\ln(t/v) + \ln\ln(t/v) + 1 - \ln 2 - \ln\ln 2]}{4v\ln 2}\right. \\ &\left. + \Theta\left(\frac{(\ln v)^2}{v^2}\right) - \Theta\left(\frac{(\ln v)\ln(t/v)}{v^2}\right) - \Theta\left(\frac{\ln\ln(t/v)}{v\ln(t/v)}\right) + \Theta\left(\left[\frac{\ln(t/v)}{v}\right]^2\right)\right\}.\end{aligned}$$

Since this is an upper limit, positive Θ terms can be dropped to give bound (98). $\diamond$

9.5 Comparison with Solution Boundary

The average number of solutions to a random CNF Satisfiability problem is

$$2^v [1 - (1-p)^v]^t \tag{99}$$

[22, eq. A.1]. This is v^n when

$$p = \frac{1}{v}\left[-\ln\left(1 - \frac{1 + \Theta(n(\ln v)/t)}{e^{(\ln 2)v/t}}\right)\right][1 - \Theta(p)]\,. \tag{100}$$

(Note that $n = 0$ corresponds to an average of one solution per problem.)

Details: Setting formula (99) to v^n, taking the t^{th} root, and rearranging gives

$$(1-p)^v = 1 - \frac{v^{n/t}}{2^{v/t}},$$

so

$$\begin{aligned} pv &= -\ln\left(1 - \frac{e^{n(\ln v)/t}}{e^{(\ln 2)v/t}}\right)[1 - \Theta(p)] \\ &= -\ln\left(1 - \frac{1 + \Theta(n(\ln v)/t)}{e^{(\ln 2)v/t}}\right)[1 - \Theta(p)]\,. \end{aligned}$$

◇

When t/v is large, this can be simplified to

$$pv = (\ln t - \ln v - \ln\ln 2)\left[1 - \Theta\left(\frac{\ln t - \ln v}{v}\right)\right] + \Theta\left(\frac{v}{t}\right) + \Theta\left(\frac{n\ln v}{v}\right). \tag{101}$$

Details:

$$\begin{aligned} pv &= -\ln\left\{1 - \left[1 + \Theta\left(\frac{n\ln v}{t}\right)\right]\left[1 - \frac{v\ln 2}{t}\left(1 - \Theta\left(\frac{v}{t}\right)\right)\right]\right\}[1 - \Theta(p)] \\ &= -\ln\left\{-\Theta\left(\frac{n\ln v}{t}\right) + \frac{v\ln 2}{t}\left[1 - \Theta\left(\frac{v}{t}\right)\right]\right\}[1 - \Theta(p)] \\ &= -\ln\left\{\frac{v\ln 2}{t}\left[1 - \Theta\left(\frac{v}{t}\right) + \Theta\left(\frac{n\ln v}{v}\right)\right]\right\}[1 - \Theta(p)] \\ &= (\ln t - \ln v - \ln\ln 2)[1 - \Theta(p)] + \Theta\left(\frac{v}{t}\right) + \Theta\left(\frac{n\ln v}{v}\right). \end{aligned}$$

Thus

$$pv = (\ln t - \ln v - \ln\ln 2)\left[1 - \Theta\left(\frac{\ln t - \ln v}{v}\right)\right] + \Theta\left(\frac{v}{t}\right) + \Theta\left(\frac{n\ln v}{v}\right).$$

◇

Note that for large t/v, the large p boundary for Probe Order Backtracking being fast (based on the upper bound analysis) is only slightly above the boundary for the number of solutions per problem being above 1. That is, the leading terms in eq. (94) are bigger than those in eq. (101) only by the amount $\ln v$. When t/v is not large, the relative distance between the two curves increases.

9.6 Intersection with Franco's Analysis

Franco gives an algorithm, [8], which makes selective use of resolution. This algorithm has the fastest proven average time for small t so long as p is not too large. Combining Franco's algorithm with Iwama's algorithm [12] gives an algorithm that is fast for all p when

$$t \le O(n^{1/3}(v/\ln v)^{1/6}). \tag{102}$$

The running time for Franco's algorithm is no more than

$$3 + v + e^{-te^{-2p(1+p)v} + (\ln 2)[8(pt)^3 v + 1]}, \tag{103}$$

[8, pp 1123–1124]. To find the intersection with the upper bound analysis of Probe Order Backtracking, we use bound (94) in Franco's bound (103) to eliminate p from the expression for the intersection point.

$$3 + v + \exp\left\{ -\left(\frac{n-1}{t\ln t}\right)^{\Theta(1)} + \frac{(8\ln 2)t^3(\ln t)^3}{v^2}\left[1 + \frac{2\ln\ln t}{\ln t} - \frac{\ln\ln v}{\ln t} - \Theta\left(\frac{\ln(n-1)}{\ln t}\right)\right]^3 + \ln 2 \right\}. \tag{104}$$

Details: Eq. (94) can be written as

$$pv \geq \ln t + 2\ln\ln t - \ln\ln v - \Theta(\ln(n-1)) = \ln\left[\frac{t(\ln t)^2}{\Theta(n-1)\ln v}\right].$$

When pv is equal to the bound

$$e^{-2p(1+p)v} = e^{-(2+2p)pv} = \left[\frac{(n-1)^{\Theta(1)}\ln v}{t(\ln t)^2}\right]^{2+\Theta(p)}.$$

When v and t are polynomially related $\ln t = \Theta(\ln v)$ so we may write this as

$$e^{-2p(1+p)v} = \frac{1}{t^{2+\Theta(p)}}\left[\frac{(n-1)^{\Theta(1)}}{\Theta(1)\ln t}\right]^{2+\Theta(p)} = \frac{1}{t^{2+\Theta(1)}}\left(\frac{n-1}{\ln t}\right)^{\Theta(1)}.$$

Plugging this and the value for p into Franco's bound gives eq. (104).$\diamond$

The bound (104) is no more than v^n when

$$t = \frac{3n^{1/3}v^{2/3}}{4(\ln 2)^{1/3}(\ln v)^{2/3}}\left[1 - \Theta\left(\frac{\ln\ln v}{\ln v}\right)\right]. \tag{105}$$

Details: Setting eq. (104) to v^n and taking logarithms gives

$$n\ln v - \Theta\left(\frac{1}{v^{n-1}}\right) = -\left(\frac{n-1}{t\ln t}\right)^{\Theta(1)} + \frac{(8\ln 2)t^3(\ln t)^3}{v^2}\left[1 + \frac{2\ln\ln t}{\ln t} - \frac{\ln\ln v}{\ln t} - \Theta\left(\frac{\ln(n-1)}{\ln t}\right)\right]^3 + \ln 2,$$

$$\frac{8(\ln 2)t^3(\ln t)^3}{v^2}\left[1 + \frac{2\ln\ln t}{\ln t} - \frac{\ln\ln v}{\ln t} - \Theta\left(\frac{\ln(n-1)}{\ln t}\right)\right]^3 = n\ln v - \Theta(1),$$

$$t^3(\ln t)^3 = \frac{nv^2\ln v}{8\ln 2}\left[1 - \Theta\left(\frac{1}{\ln v}\right)\right]\left[1 + \frac{2\ln\ln t}{\ln t} - \frac{\ln\ln v}{\ln t} - \Theta\left(\frac{\ln(n-1)}{\ln t}\right)\right]^{-3}.$$

We may write

$$\begin{aligned} t\ln t &= \frac{(n\ln v)^{1/3}v^{2/3}}{2(\ln 2)^{1/3}}\left[1 - \Theta\left(\frac{1}{\ln v}\right)\right]\left[1 + \frac{2\ln\ln t}{\ln t} - \frac{\ln\ln v}{\ln t} - \Theta\left(\frac{1}{\ln t}\right)\right]^{-1} \\ &= \frac{(n\ln v)^{1/3}v^{2/3}}{2(\ln 2)^{1/3}}\left[1 - \Theta\left(\frac{1}{\ln v}\right)\right]\left[1 - \frac{2\ln\ln t}{\ln t} + \frac{\ln\ln v}{\ln t} + \Theta\left(\frac{1}{\ln t}\right)\right] \\ &= \frac{(n\ln v)^{1/3}v^{2/3}}{2(\ln 2)^{1/3}}\left[1 - \frac{2\ln\ln t}{\ln t} + \frac{\ln\ln v}{\ln t} + \Theta\left(\frac{1}{\ln t}\right) - \Theta\left(\frac{1}{\ln v}\right)\right]. \end{aligned}$$

Define

$$y = \frac{(n\ln v)^{1/3}v^{2/3}}{2(\ln 2)^{1/3}}\left[1 - \frac{2\ln\ln t}{\ln t} + \frac{\ln\ln v}{\ln t} + \Theta\left(\frac{1}{\ln t}\right) - \Theta\left(\frac{1}{\ln v}\right)\right]$$

and solve $y = t \ln t$ for t. Consider the test solution

$$t = \frac{y}{\ln y}\left[1 + \frac{\ln\ln y}{\ln y} + a\left(\frac{\ln\ln y}{\ln y}\right)^2\right].$$

Plugging this in gives

$$\begin{aligned}
y &= \frac{y}{\ln y}\left[1 + \frac{\ln\ln y}{\ln y} + a\left(\frac{\ln\ln y}{\ln y}\right)^2\right] \ln\left\{\frac{y}{\ln y}\left[1 + \frac{\ln\ln y}{\ln y} + a\left(\frac{\ln\ln y}{\ln y}\right)^2\right]\right\} \\
&= y\left[1 + \frac{\ln\ln y}{\ln y} + a\left(\frac{\ln\ln y}{\ln y}\right)^2\right]\left[1 - \frac{\ln\ln y}{\ln y} + \frac{\ln\ln y}{(\ln y)^2} + \Theta\left(\frac{(a-1/2)(\ln\ln y)^2}{(\ln y)^3}\right)\right] \\
&= y\left[1 + (a-1)\left(\frac{\ln\ln y}{\ln y}\right)^2 + \Theta\left(\frac{\ln\ln y}{(\ln y)^2}\right)\right].
\end{aligned}$$

In the limit the left side is larger for $a < 1$ and the right side is larger for $a > 1$, so a solution is

$$t = \frac{y}{\ln y}\left[1 + \frac{\ln\ln y}{\ln y} + \Theta\left(\frac{(\ln\ln y)^2}{(\ln y)^2}\right)\right].$$

We have

$$\ln y = \frac{2\ln v}{3} + \frac{\ln\ln v}{3} + \Theta(\ln n) = \frac{2\ln v}{3}\left[1 + \frac{\ln\ln v}{2\ln v} + \Theta\left(\frac{\ln n}{\ln v}\right)\right]$$

and

$$\ln\ln y = \ln\ln v + \Theta(1),$$

so

$$\begin{aligned}
t &= \frac{3(n\ln v)^{1/3}v^{2/3}}{4(\ln 2)^{1/3}\ln v}\,\frac{1 - (2\ln\ln t)/\ln t + (\ln\ln v)/\ln t + \Theta(1/\ln t) - \Theta(1/\ln v)}{1 + (\ln\ln v)/(2\ln v) + \Theta((\ln n)/\ln v)} \\
&\qquad \times\left[1 + \frac{3\ln\ln v + \Theta(1)}{2\ln v + \Theta(\ln\ln v)} + \Theta\left(\frac{(\ln\ln v)^2}{(\ln v)^2}\right)\right] \\
&= \frac{3(n\ln v)^{1/3}v^{2/3}}{4(\ln 2)^{1/3}\ln v}\left[1 - \frac{2\ln\ln t}{\ln t} + \frac{\ln\ln v}{2\ln t} + \Theta\left(\frac{1}{\ln t}\right) - \Theta\left(\frac{1+\ln n}{\ln v}\right)\right] \\
&\qquad \times\left[1 + \frac{3\ln\ln v}{2\ln v} + \Theta\left(\frac{1}{\ln v}\right)\right] \\
&= \frac{3(n\ln v)^{1/3}v^{2/3}}{4(\ln 2)^{1/3}\ln v}\left[1 - \frac{2\ln\ln t}{\ln t} + \frac{\ln\ln v}{2\ln t} + \frac{3\ln\ln v}{2\ln v} + \Theta\left(\frac{1}{\ln t}\right) - \Theta\left(\frac{1+\ln n}{\ln v}\right)\right].
\end{aligned}$$

Since t is approximately $v^{2/3}$,

$$\frac{-2\ln\ln t}{\ln t} + \frac{\ln\ln v}{2\ln t} + \frac{3\ln\ln v}{2\ln v}$$

is approximately $-(3/4)(\ln\ln v)/\ln v$. Therefore the value of t reduces to eq. (105). $\diamond$

References

1. Cynthia A. Brown and Paul W. Purdom, *An Average Time Analysis of Backtracking*, SIAM J. Comput. **10** (1981) pp 583–593.
2. Khaled Bugrara and Cynthia Brown, *The Average Case Analysis of Some Satisfiability Model Problems*, Information Sciences **40** (1986) pp 21–38.
3. Khaled Bugrara and Paul Purdom, *Average Time Analysis of Clause Order Backtracking*, SIAM J. Comput. **23** (1993) pp 303–317.
4. Khaled Bugrara, Youfang Pan, and Paul Purdom, *Exponential Average Time for the Pure Literal Rule*, SIAM J. Comput. **18** (1988) pp 409–418.

5. Michael Buro and Hans Kleine Büning, *Report on a SAT Competition*, Bulletin of the European Association for Theoretical Computer Science **49** (1993) pp 143–151.
6. John Franco, *On the Probabilistic Performance of Algorithms for the Satisfiability Problem*, Information Processing Letters **18** (1986) pp 103–106.
7. John Franco, *On the Occurrence of Null Clauses in Random Instances of Satisfiability*, Indiana University Computer Science Tech. Report 291 (1989).
8. John Franco, *Elimination of Infrequent Variables Improves Average Case Performance of Satisfiability Algorithms*, SIAM J. Comput. **20** (1991) pp 1119–1127.
9. Allen Goldberg, *Average Case Complexity of the Satisfiability Problem*, Proc. Fourth Workshop on Automated Deduction (1979) pp 1–6.
10. Allen Goldberg, Paul Purdom, and Cynthia Brown, *Average Time Analysis of Simplified Davis-Putnum Procedures*, Information Processing Letters **15** (1982) pp 72–75. Printer errors corrected in **16** (1983) p 213.
11. G. Neil Haven, *unpublished analysis.*
12. Kazuo Iwama, *CNF Satisfiability Test by Counting and Polynomial Average Time*, SIAM J. Comput. **18** (1989) pp 385–391.
13. K. J. Lieberherr and E. Specker, *Complexity of Partial Satisfaction*, J. ACM **28** (1981) pp 411–421.
14. Henri M. Méjean, Henri Morel, Gérard Reynaud, "A Variational Method for Analysing Unit Clause Search", submitted for publication.
15. Allen Newell and H. A. Simon, "GPS, A Program that Simulates Human Thought", *Computers and Thought* (1963) pp 279–296, Edward A. Feigenbaum and Julian Feldman eds.
16. Paul W. Purdom, *Search Rearrangement Backtracking and Polynomial Average Time*, Artificial Intelligence **21** (1983) pp 117–133.
17. Paul W. Purdom, *A Survey of Average Time Analyses of Satisfiability Algorithms*, Journal of Information Processing, **13** (1990) pp 449–455. An earlier version appeared as *Random Satisfiability Problems*, Proc. of the International Workshop on Discrete Algorithms and Complexity, The Institute of Electronics, Information and Communication Engineers, Tokyo (1989) pp 253–259.
18. Paul W. Purdom, *Average Time for the Full Pure Literal Rule*, Information Sciences, to appear.
19. Paul W. Purdom, *unpublished analysis.*
20. Paul W. Purdom and Cynthia A. Brown, *An Analysis of Backtracking with Search Rearrangement*, SIAM J. Comput. **12** (1983) pp 717–733.
21. Paul W. Purdom and Cynthia A. Brown, *The Pure Literal Rule and Polynomial Average Time*, SIAM J. Comput. **14** (1985) pp 943–953.
22. Paul W. Purdom and Cynthia A. Brown, *Polynomial-Average-Time Satisfiability Problems*, Information Sciences **41** (1987) pp 23–42.
23. Paul Walton Purdom Jr. and G. Neil Haven, *Backtracking and Probing*, Indiana University Computer Science Technical Report No. 387 (1993).
24. Rok Sosič and Jun Gu, *Fast Search Algorithms for the N-Queens Problem*, IEEE Trans. on Systems, Man, and Cybernetics **21** (1991) pp 1572–1576.