TECHNICAL REPORT NO. 270

Backtracking and Random Constraint Satisfaction

by

Paul W. Purdom

Revised: November 1990

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY
Bloomington, Indiana 47405-4101

Backtracking and Random Constraint Satisfaction†

Paul Walton Purdom
Computer Science
Indiana University
Bloomington, IN 47405
U. S. A.

Abstract: The average running time used by backtracking on random constraint satisfaction problems is studied. This time is polynomial when the ratio of constraints to variables is large, and it is exponential when the ratio is small. When the number of variables goes to infinity, whether the average time is exponential or polynomial depends on the number of variables per constraint, the number of values per variable, and the probability that a random setting of variables satisfies a constraint. A method for computing the curve that separates polynomial from exponential time and several methods for approximating the curve are given. The version of backtracking studied finds all solutions to a problem, so the running time is exponential when the number of solutions per problem is exponential. For small values of the probability, the curve that separates exponential and polynomial average running time coincides with the curve that separates an exponential average number of solutions from the a polynomial number. For larger probabilities the two curves diverge. Random problems similar to those that arise in understanding line drawings with shadows require a time that is mildly exponential when they are solved by simple backtracking. Slightly more sophisticated algorithms (such as the constraint satisfaction algorithm that is used in practice) should be able to solve these rapidly.

†This work was supported by Indiana University and FAW (Research Institute for Applied Knowledge Processing at the University of Ulm).

A constraint satisfaction problem instance consists of a set of relations defined over a set of discrete variables. The goal is to find a value of each variable such that every relation evaluates to *true*. There are many applications of this type of problem. One famous application is that of understanding line drawings [6].

This class of problems includes the classical problem of satisfiability of a Boolean expression. It is NP-complete even when restricted to include only problems where the relations are functions of three variables with two values per variable or when restricted to include only problems where the relations are functions of two variables with three values per variable [5]. All known algorithms have exponential worst-case running time. There are, however, simple algorithms that are sometimes fast in practice.

Many people have measured the speed of algorithms for the constraint satisfaction problem and their work is summarized by Nudel [2]. Several people have also calculated the speed of constraint satisfaction algorithms for random problems. Haralick and Elliot [1] computed the average speed for backtracking and forward checking algorithms to solve simple random sets of problems. Nudel [2] analyzed the speed of these algorithms for more complex random problem sets.

The algorithms that have been analyzed are simpler (and almost certainly slower) than those used in practice. Waltz [6] used constraint propagation for understanding line drawings. Constraint propagation is a sophisticated form of backtracking. There are many problems for which it is fast but the simple form of backtracking considered in this paper is slow. There are very few problems where the reverse is true.

This paper finds the ratio of the number constraints to the number of variables that results in random problems being on the borderline between exponential and polynomial average time. This is the first asymptotic analysis of the average speed of backtracking for solving random constraint satisfaction problems. These results were present in preliminary form at the 1988 workshop on Mathematical and Artificial Intelligence at Ulm.

Backtracking

The backtracking algorithm consists of first checking whether there is any relation that can not be satisfied. If so, the problem has no solution. Otherwise, a variable is selected, and each value for that variable is tried. Trying a value consists of simplifying each relation to reflect the value of the variable. The simplified problem is solved by recursive application of the algorithm. The algorithm returns "Unsatisfiable" if the initial problem contains an unsatisfiable relation or if all the subproblems are unsatisfiable. It returns "Satisfiable" if any subproblem is satisfiable or if the problem contains no variables. Although a constraint satisfaction algorithm could stop as soon as one way is found to satisfy the relations, this paper analyzes a version that finds all solutions. As a result, those random problem sets where the average number of solutions is exponential require exponential average time even if the algorithm finds each solution quickly.

Backtracking. Input: A set of variables $x_1, x_2, \ldots, x_v$, sets $X_1, X_2, \ldots, X_v$ of possible values for variables $x_1, x_2, \ldots, x_v$ respectively, and a set of relations $R_1, R_2, \ldots, R_t$ of relations defined on the variables. Output: *Satisfiable* or *Unsatisfiable*.

1. If some relation is *false* for all combination of values for the variables that it depends on then return *Unsatisfiable*.
2. If the set of variables is *empty*, then return *Satisfiable*.
3. Remove x_1 from the set of variables and X_1 from the set of possible values. For each value in X_1 generate a new problem by simplying each R_i that depends on x_1. Solve each new problem

by a recursive call to this procedure. If the result of any new problem is *Satisfiable*, then return *Satisfiable*, otherwise return *Unsatisfiable.*

With some froms of problem representation and with constraints that depend on a large number of variables, Step 1 can require a large amount of time. A more primitive form of backtracking uses the following step in place of Step 1:

1′. If some relation is *false* for the current setting of the variables then return *Unsatisfiable.*

When Step 1′ is used, there is no need to simplify relations in Step 3. It is clear that the primitive version of the algorithm never generates a smaller backtrack tree, but in some cases it may be faster due to spending less time per node. The paper concentrates on the more sophisticated form of the algorithm, but it is easy to adapt the results to obtain an analysis of the more primitive form.

Random Problems

To evaluate the speed of backtracking, the following random problem set is used. Each problem in the set is formed from v variables, where each variable has $u \geq 2$ possible values. Each problem contains t relations, and each relation depends on $k \geq 1$ of the v variables. A random problem is formed by selecting with repetition t random relations. A random relation is formed by

1. selecting without repetition k of v variables and
2. for each of the u^k assignments of values of the k variables define the relation to be *true* with probability p.

Call each setting of the variables of a relation a *cell.* Thus, a random relation depends on k distinct variables, and for each cell the relation has the value *true* with probability p independent of the value for other cells.

Exact analysis

The probability that all the cells of a relation are *false* is $(1-p)^{u^k}$. The probability that some cell is *true* is $1-(1-p)^{u^k}$. The probability that every relation has some true cell is $[1-(1-p)^{u^k}]^t$. The probability that some relation has no true cell is

$$1-[1-(1-p)^{u^k}]^t, \tag{1}$$

Which is also the probability that the algorithm terminates without generating any subproblems.

Suppose i of the v variables are set (have been assigned values). In a single random relation, each variable has probability i/v of being set. The probability that the first j variables in a random relation are set is $i(i-1)\cdots(i-j+1)/[v(v-1)\cdots(v-j+1)]$. The probability that the first j are set and the remaining $k-j$ are unset is $i(i-1)\cdots(i-j+1)(v-i)(v-i-1)\cdots(v-k+j+1)/[v(v-1)\cdots(v-k+1)]$. Thus, the probability that j of the k variables of a random relation are set is

$$\binom{k}{j}\binom{v-k}{i-j}\Big/\binom{v}{i}. \tag{2}$$

If a random relation on k variables has j of its variables set then the simplified relation is a random relation on $k-j$ variables, where a random relation on zero variables is *true* with probability p and *false* with probability $1-p$.

After i variables are set, the probability that a random relation has at least one true cell is

$$g(i) = \sum_j \binom{k}{j}\binom{v-k}{i-j}\left[1-(1-p)^{u^{k-j}}\right] \Big/ \binom{v}{i}. \tag{3}$$

(The function g also depends on v, p, u and k, but dependence on these parameters is not displayed.) The probability that t random relations all have at least one true cell (after i variables have been set) is $[g(i)]^t$. This is also the probability that a particular branch of the search tree will extend to level $i+1$.

The search tree has one root node. Each extension of a node in the search tree results in u additional nodes. On level i of the search tree, there are u^i settings of the variables to consider. The average number of nodes in the entire search tree for a random problem is

$$1 + u \sum_{0 \le i < v} u^i \left[g(i)\right]^t. \tag{4}$$

The average number of solutions for a random problem is the number of variable settings times the probability that a setting makes all the relations *true*:

$$u^v[g(v)]^t = u^v p^t. \tag{5}$$

The average number of nodes generated by the *primitive form of backtracking* is given by eq. 4 with the definition of g (eq. 3) modified by retaining only the $j = k$ term in the sum. It is straightforward to modify the following analysis for this change in definition of g, so it will not be considered further. We now continue with the analysis of the more sophisticated version of backtracking.

Simple approximations

From the general form of eqs. 3 and 4, it is evident that the average number of nodes increases with v and p and it decreases with u and t.

The function $g(i)$ is particularly important in determining whether the average running time is exponential or polynomial. Let

$$p_j = 1 - (1-p)^{u^j}, \tag{6}$$

which is a increasing function of j. Note that $p_0 = p$. Since $g(i)$ is an average of the p_j (where the weights are nonnegative functions of i),

$$p_0 \le g(i) \le p_k. \tag{7}$$

Plugging these bounds into (4) gives an average number of nodes of

$$1 + u\frac{u^v - 1}{u - 1}x^t \tag{8}$$

for some x in the range $p \le x \le p_k$. The time is polynomial (less than v^n for some constant $n < \infty$) when $u^v p_k^t \le v^n$, which is the same as

$$\frac{t}{v \ln u} \ge \frac{1}{-\ln p_k}\left(1 - \frac{n \ln v}{v \ln u}\right). \tag{9}$$

The time is exponential (greater than $e^{\epsilon v}$ for some constant $\epsilon > 0$) when $u^v p_k^t \geq e^{\epsilon v}$, which is the same as

$$\frac{t}{v \ln u} \leq \frac{1}{-\ln p_k} \left(1 - \frac{\epsilon}{\ln u}\right). \tag{10}$$

Applying Taylor's theorem with a remainder to eq. 6 (using p as the variable) gives

$$p_j = u^j p (1-c)^{u^j - 1}, \tag{11}$$

for some c in the range $0 \leq c \leq p$. Setting $c = 0$ gives an upper limit of $u^j p$ for p_j and $p_0 = p$ is a lower limit, so

$$p \leq p_j \leq u^j p. \tag{12}$$

Applying the upper limit for p_k shows that the simple boundaries for polynomial and exponential time (9 and 10) are close together when the logarithm of p is much less than the logarithm of u^{-k}.

Basic asymptotics

Assume that that t is functions of v, and that k, p, and u are constant. (Letting these parameters vary slowly with v would not change the analysis much, but it would mean that more care would be needed with eq. 16, for example.)

The binomials containing i from eq. 3 can be written as

$$\binom{v-k}{i-j} \Big/ \binom{v}{i} = \prod_{0 \leq x < k-j} \left(1 - \frac{i-x}{v-x}\right) \prod_{0 \leq x < j} \left(\frac{i-x}{v-k+j-x}\right). \tag{13}$$

Since

$$1 - \frac{i-x}{v-x} = 1 - \frac{i}{v} - O\left(\frac{x}{v}\right) \tag{14}$$

and

$$\frac{i-x}{v-k+j-x} = \frac{i}{v} - O\left(\frac{x}{v}\right), \tag{15}$$

$$g(i) = \sum_j \binom{k}{j} \left(1 - \frac{i}{v}\right)^{k-j} \left(\frac{i}{v}\right)^j p_{k-j} \left[1 - O\left(\frac{k}{v}\right)\right]. \tag{16}$$

Define $f(x)$ by

$$f(x) = \sum_j \binom{k}{j} (1-x)^{k-j} x^j p_{k-j}. \tag{17}$$

Then, the size of the search tree is given by

$$1 + u \sum_{0 \leq i < v} u^i \left[f\left(\frac{i}{v}\right)\right]^t O(e^{kt/v}). \tag{18}$$

The rest of the paper considers the conditions where the average time is a polynomial or exponential function of v. This is done by studying eq. 18 with the big O term dropped. Thus, we are assuming that kt/v is bounded by a constant times $\ln v$ as v goes to infinity. (When k, p, and u are fixed the boundary between polynomial and exponential time always occurs at a finite value of t/v, so the assumption of a bound on kt/v is not actually used.)

Numerical bounds

If M is the maximum term in the sum from eq. 18, then the number of nodes is at most $1+uvM$. The number of nodes is at least $1+uM$. The maximum term can be found by setting to zero the derivative (with respect to $x = i/v$) of the logarithm of the summand. If the derivative is negative for all x between 0 and $1-1/v$, then $x = 1-1/v$ maximizes the summand. (The derivative is never positive for the entire range.) Let x be the value of i/v that maximizes the summand. The summand is maximized by the root of

$$\ln u + \frac{t}{v}\frac{f'(x)}{f(x)} = 0, \tag{19}$$

when the root is in the range $0 \le x \le 1 - 1/v$, and by $x = 1 - 1/v$ when the root is greater than $1 - 1/v$.

The terms in the summand require i to be an integer while the value of x that is the root of eq. 19 may be noninteger. This never causes a problem in determining where the boundary between polynomial and exponential time is, because no level of the backtrack tree has more than u times as many nodes as the previous level. Thus, reducing x slightly from the root value never has an exponential effect on the number of node. Also, we can increase the upper limit on x to 1 (instead of $1 - 1/v$) without affecting our final conclusion. The average time is polynomial when $vu^{x+1}[f(x)]^t \le v^n$, which gives

$$\frac{x}{v}\ln u + \frac{t}{v}\ln f(x) \le \frac{(n-1)\ln v - \ln u}{v}. \tag{20}$$

In the limit as v goes to infinity, the right side is zero. The average time is exponential when

$$\frac{x}{v}\ln u + \frac{t}{v}\ln f(x) \ge \frac{\epsilon v - \ln u}{v}. \tag{21}$$

The right side can be as close to zero as we like by choosing ϵ be small. The boundary between polynomial and exponential time is given by

$$x\ln u + \frac{t}{v}\ln f(x) = 0. \tag{22}$$

As v goes to infinity, if the limit of the left side is less than zero, then the time is polynomial; if it is greater than zero, then the time is exponential; and if it is zero, then the rate at which the limit approaches zero determines the asymptotic running time.

Using eq. 22 to eliminate t/v in eq. 19 gives

$$1 - \frac{xf'(x)}{f(x)\ln f(x)} = 0 \tag{23}$$

as the equation for x, provided it has a root with $0 \le x \le 1$. Otherwise the value of x is 1. Once the value of x is known, the value of t/v is given by eq. 22, which can be written as

$$\frac{t}{v} = \frac{x\ln u}{-\ln f(x)}. \tag{24}$$

Since $f(x) < 1$, eq. 24 always gives a positive limit for t/v.

The function $f'(x)$ is

$$f'(x) = k[x^{k-1}p_0 - (1-x)^{k-1}p_k] + \sum_{1 \le j \le k-1} \binom{k}{j} x^{j-1}(1-x)^{k-j-1}(j-kx)p_{k-j}. \tag{25}$$

The function $f'(x)$ is negative, and it becomes more negative as x increases (the constraints become more effective as more variables are set). The extreme values are $f'(0) = -k(p_k - p_{k-1})$ and $f'(1) = -k(p_1 - p_0)$.

The condition for $x = 1$ to be a root of eq. 23 is

$$1 + \frac{k(p_1 - p_0)}{p \ln p} = 0. \tag{26}$$

Using p_* as the value of p that satisfies eq. 26 and $p_1 - p_0 = 1 - p - (1-p)^u$, eq. 26 can be written as

$$p_*(k - \ln p_*) = k[1 - (1-p_*)^u]. \tag{27}$$

The left side is a decreasing function of p_* and the right side is an increasing function, so there is one solution for $0 < p_* < 1$. Expanding the right side of eq. 27 in a power series gives

$$p_* \ge e^{-k(u-1)}. \tag{28}$$

This bound results in the left side of eq. 27 being kup_* while the right side is smaller than this. When the right side of eq. 28 is small, it is a good approximation to the actual value. The solution and bound (eq. 28) for selected values of k and u are given in Table 1.

k	u	p_*	lower bound	k	u	p_*	lower bound
2	2	5.38×10^{-1}	1.35×10^{-1}	3	2	7.73×10^{-2}	4.97×10^{-2}
2	3	2.23×10^{-2}	1.83×10^{-2}	3	3	2.56×10^{-3}	2.48×10^{-3}
2	11	2.06×10^{-9}	2.06×10^{-9}	3	11	9.36×10^{-14}	9.36×10^{-14}

Table 1. The largest probability that results in the boundary for exponential average time being the same as the boundary for an exponential average number of solutions

When $p \le p_*$, the average running time is polynomial if and only if the average number of solutions per problem is polynomial, and eq. 5 implies that the boundary is

$$\frac{t}{v} = \frac{\ln u}{-\ln p}. \tag{29}$$

Power series approximation

Taylor's theorem shows that p_j can be bounded by a function of the form $\beta\alpha^j$, but it does not provide the best approximation of this form. The best such approximation depends on the fact that p_{i+1}/p_i is a decreasing function of i. To prove this start with

$$\frac{p_{i+1}}{p_i} - \frac{p_{i+2}}{p_{i+1}} \ge 0 \quad \text{if and only if} \quad p_{i+1}^2 - p_i p_{i+2} \ge 0. \tag{30}$$

Let $r = (1-p)^{u^i}$. For $0 \le p \le 1$, the range of r is $0 \le r \le 1$. In terms of r, $p_{i+1}^2 - p_i p_{i+2}$ is

$$\begin{aligned}(1-r^u)^2 - (1-r)(1-r^{u^2}) &= r[(1-r^{u-1})^2 - r^{2(u-1)}(1-r)(1-r^{(u-1)^2}] \\ &\ge r[(1-r^{u-1} - (1-r)(1-r^{(u-1)^2}].\end{aligned} \tag{31}$$

Thus eq. 30 is true for u provided it is true for $u-1$. Direction calculation shows that it is true for $u = 0$ (if $r > 0$) and for $u = 1$. By induction, it is true for all positive integer u.

Since p_{i+1}/p_i is an decreasing function of i, an upper limit on p_i of the form $\beta\alpha^i$ can be found by setting the error to zero at two consecutive points, p_j and p_{j+1}. Thus, if α_j and β_j are chosen so that

$$p_j = \beta_j \alpha_j^j \qquad p_{j+1} = \beta_j \alpha_j^{j+1}, \tag{32}$$

then

$$p_i \le \beta_j \alpha_j^i \tag{33}$$

for all i in the range $0 \le i \le k$. Solving eq. 32 for α_j and β_j gives

$$\alpha_j = \frac{p_{j+1}}{p_j}, \qquad \beta_j = p_j \alpha_j^{-j}. \tag{34}$$

Approximating p_i with $\beta_j \alpha_j^i$ results in no error at $i = j$ and at $i = j+1$. If α is chosen between α_{j-1} and α_j, and β is determined by eq. 34, then the error at p_{j-1} is reduced while the error at p_{j+1} is increased.

A lower limit on p_i of the form $\beta\alpha^i$ can be found by setting the error to zero at p_0 and p_k. This gives

$$\alpha = \left(\frac{p_k}{p_0}\right)^{1/k}, \qquad \beta = p_0. \tag{35}$$

Thus, we have both upper and lower bound approximations on p_i of the form $\beta\alpha^i$.

Approximating p_i with $\beta\alpha^i$ gives

$$f(x) = \sum_j \binom{k}{j} x^j (1-x)^{k-j} \beta\alpha^j = \beta\alpha^k [1 - (1-\alpha^{-1})x]^k. \tag{36}$$

This upper bound can be written in the form $a(1-bx)^k$ with

$$a = \beta\alpha^k, \qquad b = 1 - \alpha^{-1}. \tag{37}$$

With this approximation to $f(x)$, eq. 19 becomes

$$\ln u - \frac{bkt/v}{1-bx} = 0, \qquad x = \frac{1}{b} - k\frac{t}{v \ln u}. \tag{38}$$

If the resulting x is negative, it is replaced with zero, if it is greater than 1 it is replaced by 1. It is useful to define

$$y = bk\frac{t}{v \ln u}. \tag{39}$$

The average running time is polynomial in three cases. The best bound comes from setting t/v (and thus y) as small as possible. Therefore, the last case that applies gives the best bound.

Case 1: $x = 0$, $y \geq 1$.

The number of nodes is no more than $1 + uva^t$, which is polynomial when

$$y \ln a \leq \frac{bk[(n-1)\ln v - \ln u]}{v \ln u}. \tag{40}$$

(The algebraic details of the derivation of eq. 40 have been omitted. They are similar to those given for eq. 42.) Eq. 40 has a solution with $y \geq 1$ for large v only if

$$a \leq 1. \tag{41}$$

Since n can be chosen so that the right side is positive for large v, if $a < 1$, then any $y \geq 1$ satisfies eq. 40. The next case always gives results at least as good as this case.

Case 2: $0 \leq x < 1$, $1 - b \leq y \leq 1$.

The number of nodes is no more than $1 + vu^{1+v/b-kt/\ln u}[bkt/(v \ln u)]^{kt} a^t$, which is polynomial when

$$\begin{aligned}
e^{(v/b+1)\ln u - kt} \left(\frac{bkta^{1/k}}{v \ln u}\right)^{kt} &\leq v^{n-1} \\
\left(\frac{a^{1/k} y}{e}\right)^{kt} &\leq \frac{v^{n-1}}{e^{(v/b+1)\ln u}} \\
\left(\frac{a^{1/k} y}{e}\right)^{y} &\leq \frac{v^{(n-1)b/(v \ln u)}}{e^{1+v/b}} \\
y\left(\ln y + \frac{\ln a}{k} - 1\right) &\leq \frac{(n-1)b \ln v}{v \ln u} - 1 - \frac{b}{v} \\
y\left(1 - \ln y - \frac{\ln a}{k}\right) &\geq 1 - \frac{(n-1)b \ln v}{v \ln u} + \frac{b}{v}.
\end{aligned} \tag{42}$$

For large v the right side of eq. 42 approaches 1 and there are solutions with $y > 0$ only if the part of the left side in parentheses is positive. The smallest value of y results when the greater than or equal in eq. 42 is equal. The point $y = 1$ is a solution to the resulting equation when $a = 1$, and $y = 1 - b$ is a solution for $a = e^{-kb/(1-b)}/(1-b)$. Intermediate values of a lead to intermediate values of y, so case 2 gives the best value for y when

$$\frac{e^{-kb/(1-b)}}{1-b} \leq a \leq 1. \tag{43}$$

Eq. 42 provides a good approximation to the boundary when a is small, but the error becomes very large when a is near 1.

Case 3: $x = 1$, $y \leq 1 - b$.

The number of nodes is no more than $1 + vu^{1+v}(1-b)^{kt}a^t$, which is polynomial when

$$y \geq \frac{kb + k[(n-1)\ln v - \ln u]/(v \ln u)}{-\ln p}. \tag{44}$$

For large v eq. 44 has solutions with $y \le 1 - b$ when

$$a \le \frac{e^{-kb/(1-b)}}{1-b}. \tag{45}$$

Eq. 44 gives the same boundary as eq. 29, but eq. 27 shows that it actually applies for a larger range than that indicated by eq. 45.

When using eq. 42 for an upper bound to the boundary between polynomial and exponential time, a choice must be made for the value of α to use. Choosing α near α_0 results in a good approximation for small p, but the error becomes large when p is large enough for $\beta\alpha^k$ to be near 1, and the approximation does not apply when it is larger than 1. Smaller values of α give larger errors for small p but postpone the occurrence of large errors. Choosing $\alpha = \alpha_{k-1}$ results in $\beta\alpha^k$ being below 1 for all values of p. The sample calculations given below shows that this choice gives small error for all p when u is large.

The accuracy of the polynomial approximation can be improved by increasing the number of terms in the approximation, but the resulting formulas do not have simple solutions.

Refined simple approximations.

One disadvantage of eq. 42 is that y is defined by implicit equation of the form

$$y(1 - \ln y + z) = 1, \tag{46}$$

where $z = (-\ln a)/k > 0$. Since

$$\frac{1}{1 - \ln y + z} \tag{47}$$

is an increasing function of y, replacing the y in eq. 47 with a lower bound on y gives (an improved) lower bound on y and replacing it with an upper bound gives an upper bound. When eq. 42 applies, y must be in the range $1 - b \le y \le 1$. Using 1 as the upper limit on y, α_{k-1} for α, and β_{k-1} for β, gives polynomial average time (in the limit of large v) for

$$\frac{t}{v} > \frac{\ln u}{k(1 - p_{k-1}/p_k)[1 - (\ln p_k)/k]} \tag{48}$$

for $p > p_*$. Using $1 - b$ as lower limit on y, and α and β from eq. 35 gives exponential average time (in the limit of large v) for

$$\frac{t}{v} < \frac{\ln u}{k[1 - (p_0/p_k)^{1/k}][1 + (1 - 1/k)\ln p_k]}. \tag{49}$$

These last two results are obtained by approximating approximations. Sample calculations show that eq. 48 often gives a bound that is about twice the bound from the polynomial approximation, while eq. 49 usually gives a bound close to the polynomial approximation. However, the polynomial approximation associated with eq. 48 often has a small error while the one associated with eq. 49 often has a large error. Thus eq. 48 and eq. 49 gives bounds that are close together only for small p.

Calculations

For $k = 2$ and 3, and for $u = 2$, 3 and 11, Figures 1–6 show the boundaries computed from the previous formulas. The horizontal axis is used for t/v to facilitate comparisons with previously

published analyses of backtracking for conjunctive normal form satisfiability problems [3, 4]. When t/v is greater than (to the right of) the leftmost curve (marked with an 'S'), the average number of solutions is polynomial; when it is less than (to the left of), the average is exponential. When t/v is less than the second leftmost curve, the polynomial approximation (eqs. 42 and 44) shows that the running time is exponential. This curve is only slightly to the right of the first curve for $k = 2$, $u = 2$, but it is well to the right of it for the other values of k and u. The third leftmost curve (marked with an 'N') gives the true dividing line between polynomial and exponential average time. When t/v is to the right of the this curve the average running time is polynomial; when it is to the left, the average is exponential. These curves show that the polynomial approximation does not give a good lower bound for t/v except when p is small. The remaining curves are upper bounds on t/v obtained from the polynomial approximation, using $\alpha = \alpha_i$ for different values of i. The upper bound for $i = k - 1$ applies for all p giving the rightmost most curve that goes across the entire figure. The upper bounds for smaller i only apply for small p, giving curves that hook to the right and terminate. The larger the value of i the larger the value of p where the curve terminates.

These curves show that the choice $\alpha = \alpha_{k-1}$ gives a rather good approximation to the true boundary when u is large. The figures for $u = 11$ show the approximate boundary almost on top of the true boundary. When variables can have a large number of values, the boundary between exponential and polynomial time is determined primarily by p_k, the probability that a relation is *true* after all the variables are set, and by p_{k-1}, the probability that a relation is *true* after all but one of the variables are set. Using $\alpha = \alpha_{k-1}$ approximates these two probabilities with no error. The lower bound approximation is not so accurate. It approximates p_k with no error, but, when p is not small, it does not give a good approximation to p_{k-1}.

It is interesting to compare Waltz's figure recognition problem with these random problems. The constraints in his problem depend on between 2 and 4 variables; the case $k = 3$ is particularly common. Each variable occurs in two constraints, so $t/v = 3/2$ is appropriate for $k = 3$. The variables have 11 values, so $u = 11$. The constraints for 3 variables have a probability $p = 0.0027$ [7]. In Figure 6, the point with these parameters is in the region where the average number of solutions is small but the average running time is slightly exponential. Although machine vision problems are definitely not random constraint satisfaction problems (for example each problem has at least one solution), the analysis of random problems suggests that large problems usually have a single solution, that simple backtracking can not find the solution rapidly, and that algorithms only slightly more powerful can find the solution rapidly. The constraint propagation algorithm that Waltz used is similar to backtracking, but it is more powerful, and he was able to find solutions rapidly.

Summary of results

This section approximations to the boundary between polynomial and exponential time that are easy to state. Consult the preceding sections for methods that give greater accuracy.

The average number of solutions is polynomial for

$$\frac{t}{v} > \frac{\ln u}{-\ln p}, \tag{50}$$

and exponential for

$$\frac{t}{v} < \frac{\ln u}{-\ln p}. \tag{51}$$

For $p < p_*$ where p_* is the root of the equation

$$p_*(k - \ln p_*) = k[1 - (1 - p_*)^u], \tag{52}$$

eqs. 50 and 51 also give the conditions for the average running time being polynomial or exponential. A bound on p_* is

$$p_* \geq e^{-k(u-1)}. \tag{53}$$

When this bound is small, the actual value of p_* is close to the bound.

For $p > p_*$ the average running time is polynomial when

$$\frac{t}{v} < \frac{[1-(1-p)^{u^{k-1}}]\ln u}{(1-p)^{u^{k-1}}[1-(1-p)^{(u-1)u^{k-1}}]\{k-\ln[1-(1-p)^{u^k}]\}}. \tag{54}$$

It is exponential when

$$\frac{t}{v} < \frac{[1-(1-p)^{u^k}]^{1/k}\ln u}{\{[1-(1-p)^{u^k}]^{1/k}-p^{1/k}\}\{k-(k-1)\ln[1-(1-p)^{u^k}]\}}. \tag{55}$$

Acknowledgements. I wish to thank Bernard Nudel and Cynthia Brown for discussing this problem. I wish to thank Andrew Leonard for discussing the approximation of p_i and an anonymous referee for correcting an error in a former version of eq. (2).

References

1 R. M. Haralick and G. L. Elliot, *Increasing Tree Search Efficiency for Constraint Satisfaction Problem*, Artificial Intelligence **14** (1980), pp. 263–313.

2 Bernard Nudel, *Consistent-Labeling Problems and their Algorithms: Expected-Complexities and Theory-Based Heuristics*, Artificial Intelligence **21** (1983), pp. 135–178. Also in *Search and Heuristics*, edited by J. Pearl, North Holland, New York (1983), pp. 135–178.

3 Paul Walton Purdom Jr., *Search Rearrangement Backtracking and Polynomial Average Time*, Artificial Intelligence **21** (1983), pp. 117–133.

4 Paul Walton Purdom Jr. and Cynthia A. Brown, *Polynomial-Average-Time Satisfiability Problems*, Information Sciences, **41** (1987), pp. 23–42.

5 Paul Walton Purdom Jr., Cynthia A. Brown, and Edward L. Robertson, *Backtracking with Multi-Level Dynamic Search Rearrangement*, Acta Informatica **15** (1981), pp. 99–113.

6. David L. Waltz, *Understanding Line Drawings of Scenes with Shadows*, in *The Psychology of Computer Vision*, edited by P. H. Winston, McGraw-Hill, New York (1975), pp. 19–91.

7. Patrick Henry Winston, *Machine Vision* , in *The Psychology of Computer Vision*, edited by P. H. Winston, McGraw-Hill, New York (1975), pp. 1–17.

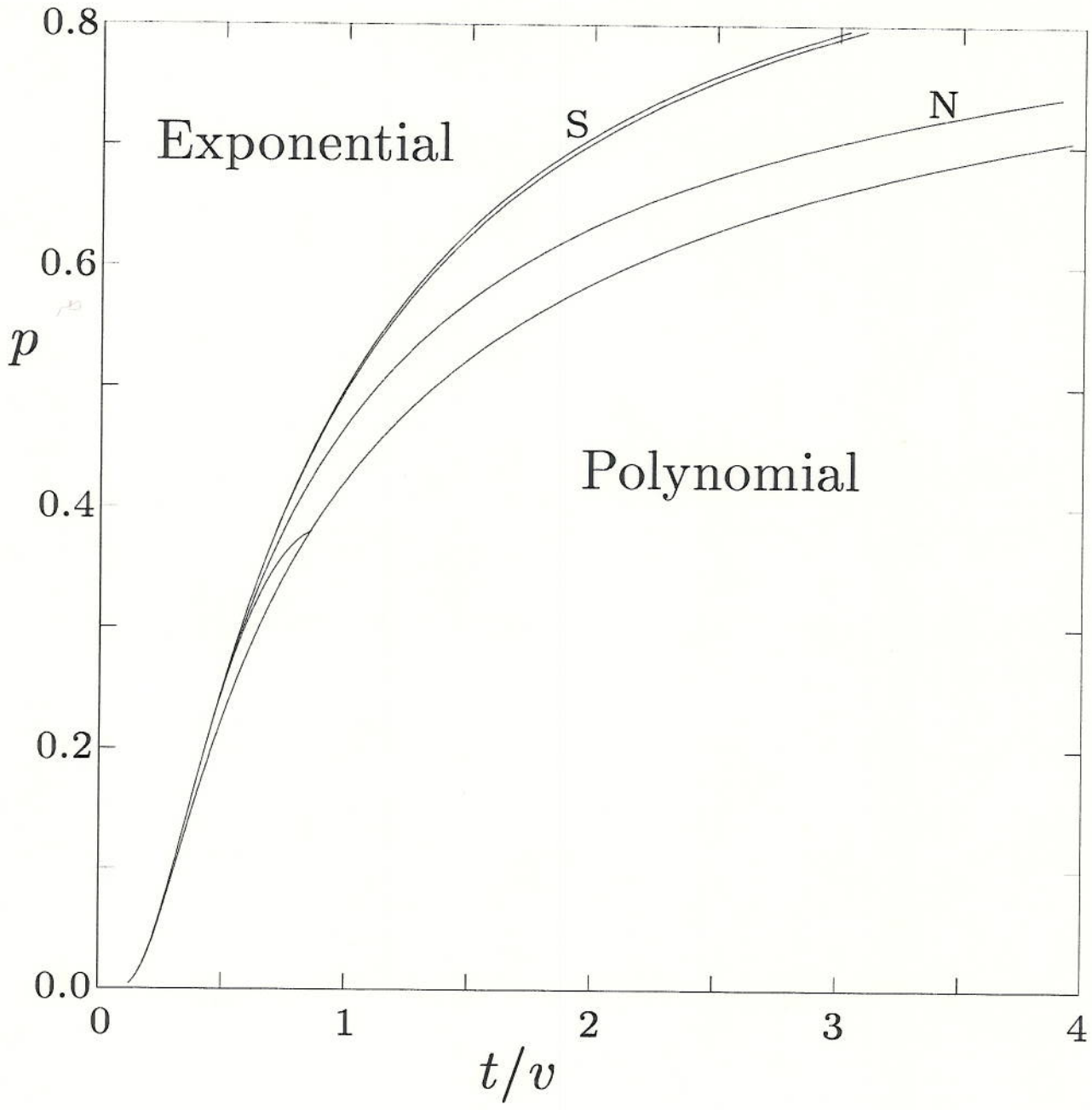

Fig. 1. Constraint satisfaction for $k = 2$ and $u = 2$. The curve marked with S shows the dividing line between an exponential and polynomial number of solutions. The curve marked with N shows the dividing line for the number of nodes. The remaining curves are the dividing lines given by upper and lower bound approximations to the N curve.

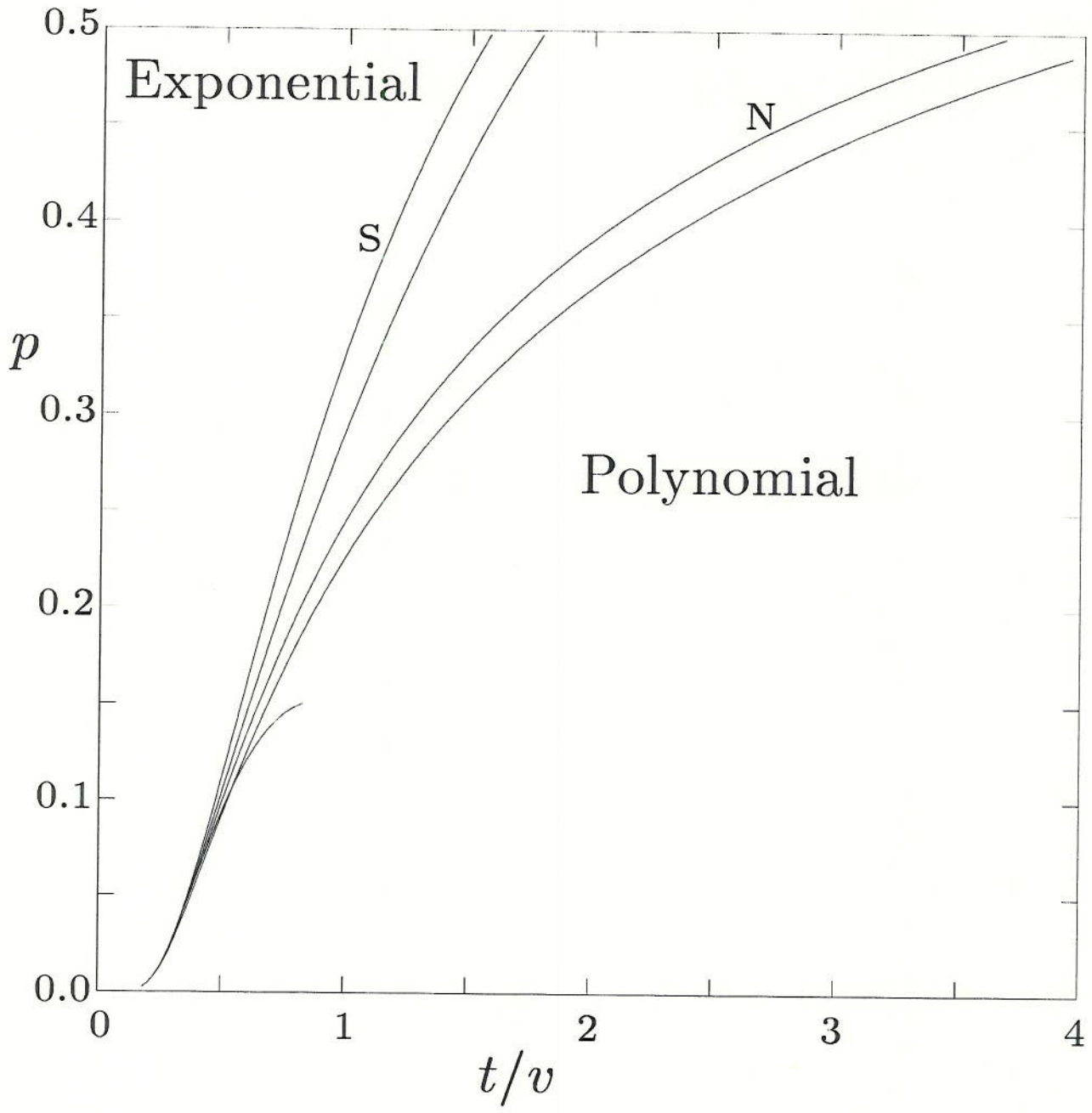

Fig. 2. Constraint satisfaction for $k = 2$ and $u = 3$.

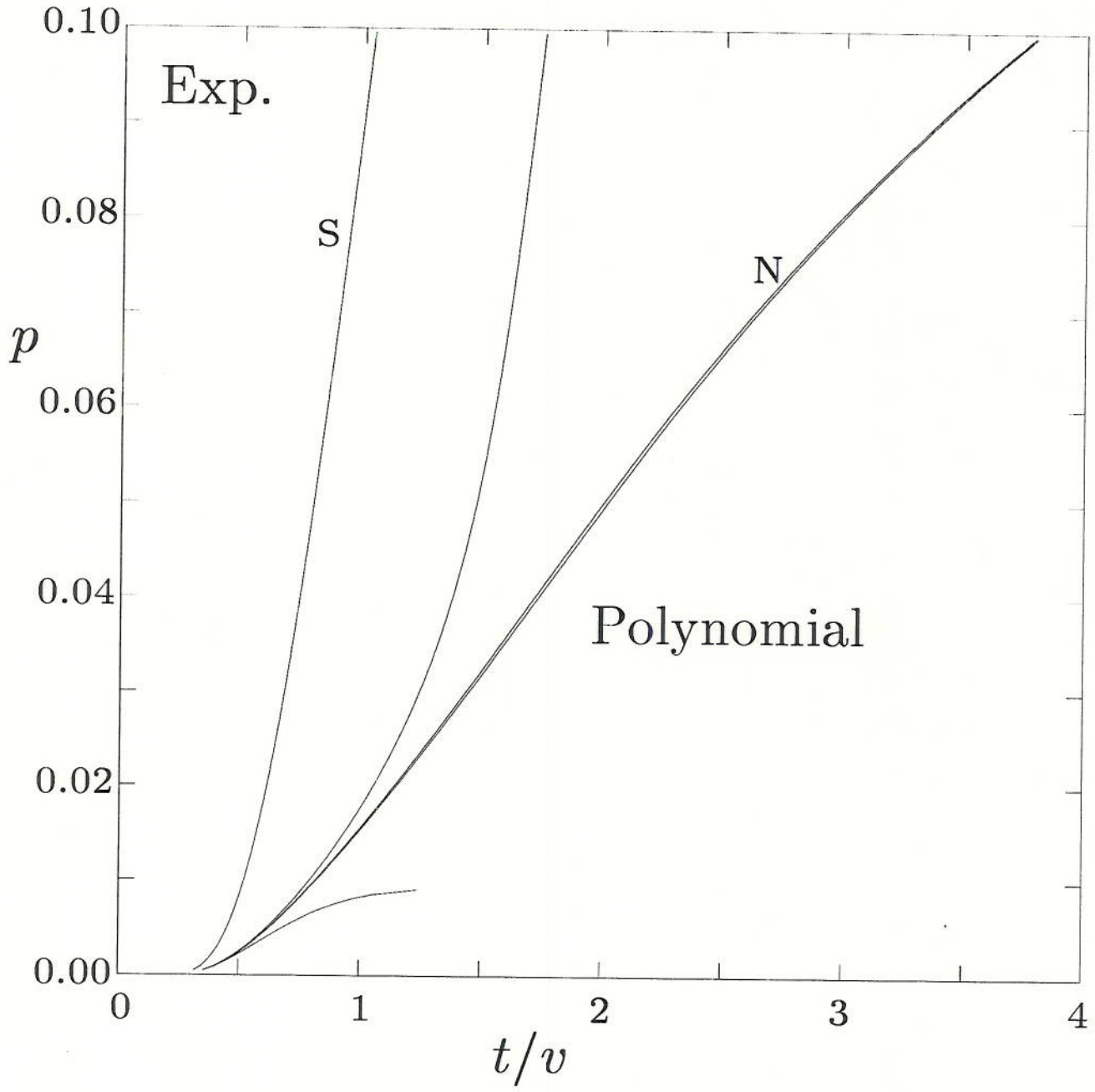

Fig. 3. Constraint satisfaction for $k = 2$ and $u = 11$.

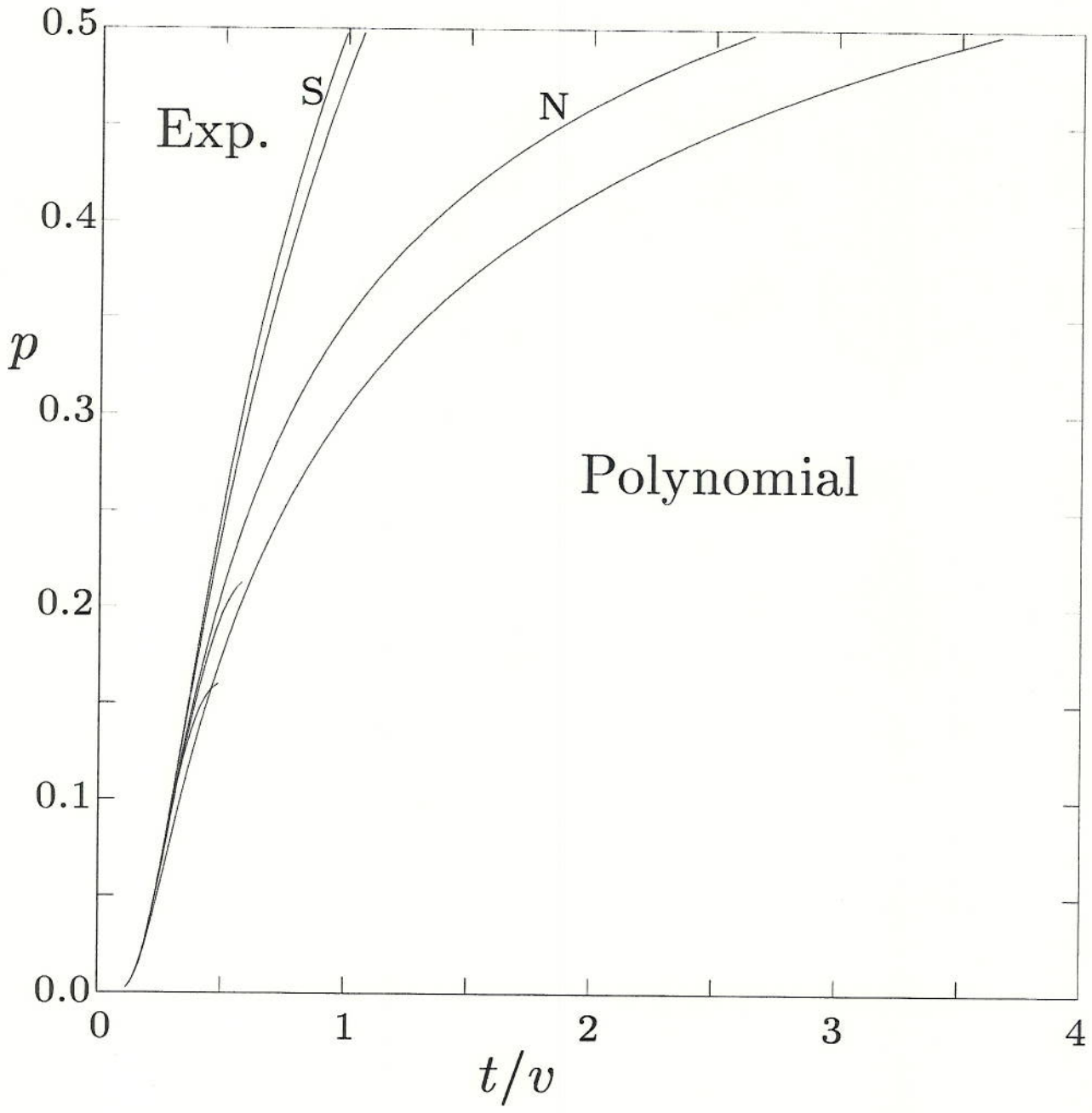

Fig. 4. Constraint satisfaction for $k = 3$ and $u = 2$.

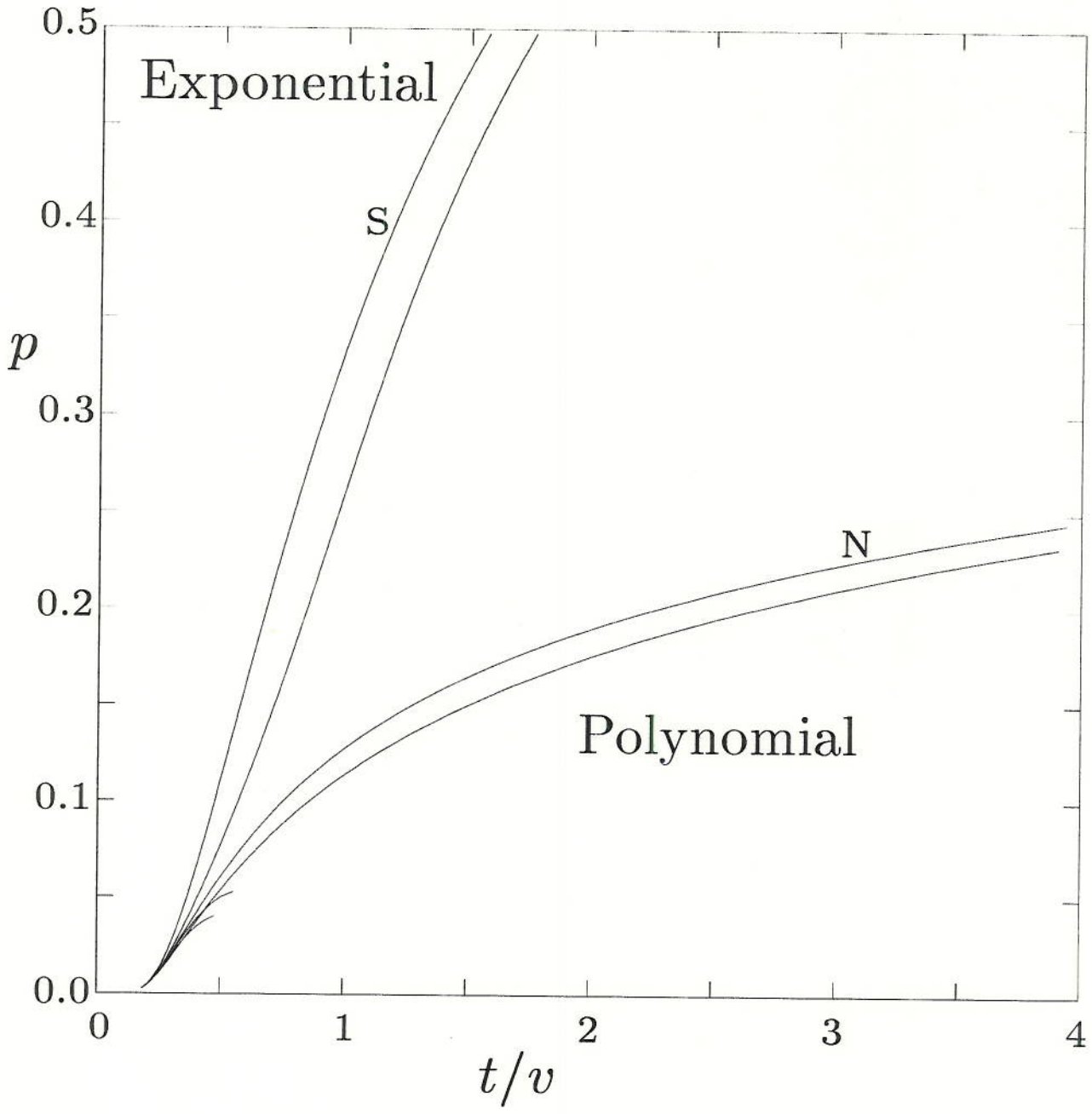

Fig. 5. Constraint satisfaction for $k = 3$ and $u = 3$.

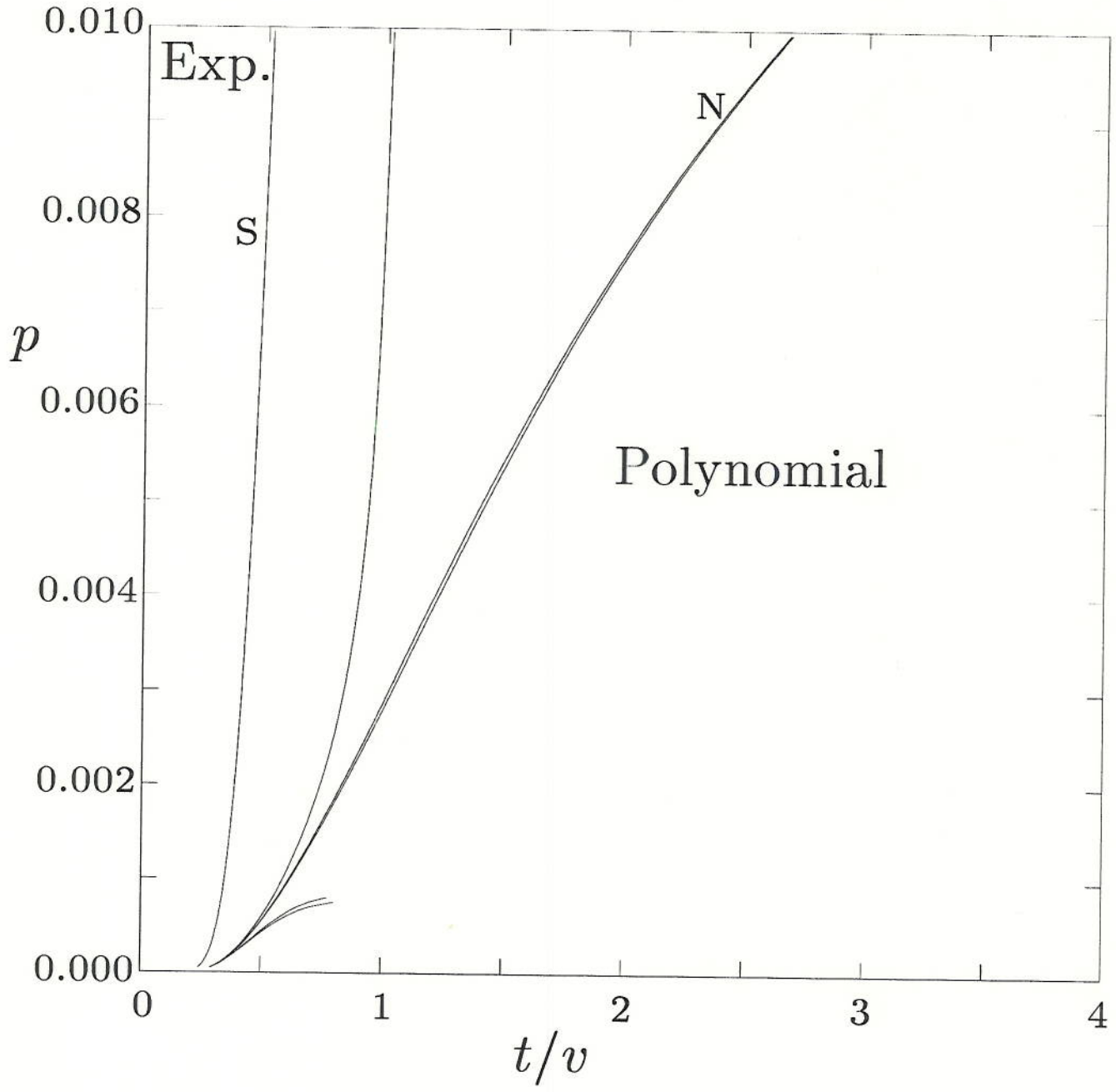

Fig. 6. Constraint satisfaction for $k = 3$ and $u = 11$.