Worst-case Time Bounds for Coloring and Satisfiability Problems

TOMÁS FEDER* RAJEEV MOTWANI†

February 8, 2002

Abstract

We consider worst case time bounds for NP-complete problems. For the $(k, 2)$-satisfiability problem, we give a randomized algorithm that runs in time $O^*((k!)^{n/k})$. This bound is $O((k/c_k)^n)$ with c_k increasing to the asymptotic limit e, and improves for $k \geq 11$ over the $O((0.4518k)^n)$ randomized bound of Eppstein [3]. A special case of $(k, 2)$-satisfiability is k-colorability; here we achieve the above time bound for a slightly larger c_k that has the same asymptotic behaviour.

*E-mail: `tomas@theory.stanford.edu`

†Department of Computer Science, Gates 4B, Stanford University, Stanford, CA 94305-9045. E-mail: `rajeev@cs.stanford.edu`. Supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, an ARO MURI Grant DAAH04-96-1-0007, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

1 Introduction

It is by now taken for granted that once a problem is shown to be NP-hard, it is extremely unlikely that it can be solved in worst-case time less than exponential in input size. Various attempts have been made to deal with this evidence of intractability of a problem. One approach is to design algorithms with polynomially bounded *average-case* running time, under the assumption that the input is drawn from some pre-specified probability distribution. Such algorithms are now viewed with disfavor since only extremely simple input distributions (such as uniform) facilitate an average-case analysis, and in any case it is not clear that average-case results truly reflect the performance of algorithms in practice. A more popular approach, applicable only to optimization problems, is to design approximation algorithms that run in polynomial time and deliver sub-optimal solutions but with some performance guarantee. It is more difficult to deal with NP-hard decision problems such as: *satisfiability*, *graph k-colorability*, or *Hamiltonian paths*. To extend the theory of approximation algorithms to such problems requires converting them to unnatural optimization problems.

Consider for example the problem of *graph k-colorability*, a problem usually motivated by pointing out its application to register allocation in compilers. Considerable effort has gone into the design of approximation algorithms for graph coloring, leading to the recent result of Karger, Motwani, and Sudan [8]. However, as pointed out by Motwani, Palem, Sarkar, and Reyen [13], register allocation requires the solution of the decision problem of whether a graph is k-colorable, or possibly determining a k-colorable subgraph, and most definitely not the optimization problem of finding a coloring of a k-colorable graph with the minimum possible number of colors (as a function of k). Thus, in the primary application of this problem the goal is to find the most efficient algorithm for the decision problem, and if that requires super-polynomial time, so be it. Typically, the graphs encountered in practice are reasonably small and it is not inconceivable that even an exponential time algorithm be implemented in this setting.

We believe that there is considerable value to the design of even exponential time algorithms for NP-hard decision problems, where the goal is to obtain running time smaller than that required for the naive approach of enumerating all possible solutions. For example, the satisfiability of a boolean formula with n variables can be determined in time $2^n \times n^{O(1)}$ in the worst-case. Here the goal should be to reduce the exponential dependence on n even if it is merely the reduction of the exponent by a constant factor. Note that reducing the exponent by a factor of 2 has tremendous value as it will double the size of input instances that can be successfully handled in practice. In fact, of late there has been considerable research into the issue of improving the exponent in the running time bound for problems such as k-satisfiability [9, 11, 14, 15, 18, 20, 21, 23], graph coloring [2, 10, 19], independent sets [6, 7, 12, 16, 17, 22], and Hamiltonian paths [1, 4, 5]. It should be noted that some algorithms for improved exponents are based on dynamic programming and require *exponential space*. For example, the algorithm of Held and Karp [5] for Hamiltonian paths has this defect and is infeasible in practice because even though practitioners may be willing to put up with exponential running time, it is unlikely that they will implement an exponential-space algorithm, preferring instead to use ad-hoc heuristics.

2 The (k, l)-Satisfiability Problem

Throughout this paper, $O^*(f(n))$ will denote $O(n^c f(n))$.

Paturi, Pudlák, and Zane [14] gave a randomized algorithm running in time $O^*(2^{(1-\frac{1}{k})n})$ for solving the k-satisfiability problem with n variables. Schöning [21] gave a randomized algorithm running in time $O^*((k/2)^n)$ for solving the $(k, 2)$-satisfiability problem. We combine ideas from these two results to give a randomized

$$O^*\left(\prod_i \min\left(\frac{k_i}{2}, k_i!^{\frac{1}{k_i}}\right)\right)$$

algorithm for $(k_i, 2)$-satisfiability. This is the problem where variable x_i has k_i possible values and constraints involve two variables at a time; they may be taken to be of the form $x_i \neq a \vee x_j \neq b$. The quantity $\frac{k}{2}$ is smaller for $k \leq 5$, while $k!^{\frac{1}{k}}$ is smaller for $k \geq 6$ and is asymptotically $\frac{k}{e}$.

When all k_i are equal to k, this gives a randomized algorithm for $(k, 2)$-satisfiability running in time $O^*((k!)^{n/k})$, thus improving over the result of Schöning for $k \geq 6$ and over the $O((0.4518k)^n)$ bound with $k > 3$ of Eppstein [3] for $k \geq 11$.

Consider the following simple algorithm for $(k_i, 2)$-satisfiability. Choose a variable x_i at random, and assign to it a value chosen at random from the k_i possible values; then restrict the sets of possible values for the remaining variables to values consistent with the chosen value for x_i; repeat the above for the resulting problem on the remaining variables. We shall show that running this algorithm $O^*(\prod_i k_i!^{\frac{1}{k_i}})$ times produces a solution with probability close to 1 if one exists.

We want to solve the $(k_i, 2)$-satisfiability problem. Let

$$\phi_k = \frac{1}{k}\sum_{i=1}^{k} \log i = \frac{1}{k}\log(k!),$$

where the log is to base 2. For a solution x, let

$$\text{value}(x) = \prod_i \frac{1}{\text{poss}(i, x)},$$

where $\text{poss}(i, x)$ is the number of possible values for variable x_i in a solution, where the remaining variables are as in solution x.

Lemma 1 *Let S be the set of satisfying solutions, and suppose S is nonempty. Then* $\sum_{x\in S} \text{value}(x) \geq 1$.

Proof: Let $x_1, x_2, \ldots, x_n$ be the variables. Construct a tree whose nodes at depth i (the root has depth 0) are the assignments to the first i variables that can be extended to solutions in S. A partial solution at depth $i - 1$ has as children the extensions of the partial solution that assign possible values for variable x_i.

Now assign numbers to the nodes of the tree. The root has number 1. If a node has number r, and has d children, its children have number r/d. Thus the leaves have number

$1/(d_1 d_2 \cdots d_n)$, where d_i is the number of children of the node at depth $i-1$ in the path from the root to the leaf. Since the number at a node is the sum of the numbers at its children, the numbers in the leaves sum to 1. For a solution x at a leaf, we have $d_i \geq \mathrm{poss}(i,x)$, so $\mathrm{value}(x) \geq 1/(d_1 d_2 \cdots d_n)$ and therefore $\sum_{x\in S} \mathrm{value}(x) \geq 1$. ■

Describe a solution x with respect to a random permutation of the variables, by choosing the value for each variable out of the values consistent with the values of variables that precede it in the permutation.

Lemma 2 *A solution x can be encoded with a number of bits whose expected value is*

$$\sum_i \frac{1}{k_i + 1 - \mathrm{poss}(i,x)} \sum_{j=\mathrm{poss}(i,x)}^{k_i} \log j \leq \sum_i \phi_{k_i} - \log \mathrm{value}(x).$$

Proof: Suppose first that the $k - \mathrm{poss}(i,x)$ forbidden values for variable x_i are each forbidden by a clause for a different variable x_j. Then in the ordering of these $k+1-\mathrm{poss}(i,x)$ variables, variable i may fall with equal probability in any position, proving the

$$\frac{1}{k+1-\mathrm{poss}(i,x)} \sum_{j=\mathrm{poss}(i,x)}^{k} \log j$$

bound. Note that by an averaging argument this quantity is at most

$$\frac{1}{k}\sum_{j=1}^{k} \log(j-1+\mathrm{poss}(i,x)) \leq (k^{-1}\sum_{j=1}^{k}\log j) + \log \mathrm{poss}(i,x),$$

proving the second bound.

It remains to argue that the case where the $k-\mathrm{poss}(i,x)$ forbidden values are forbidden one at a time is indeed the worst case. Suppose that we have a different case, and consider the variable x_j that forbids the largest number c of values for variable x_i. Insert variable x_j last in the choice of a random permutation of the variables. Thus, before x_j is inserted, we have $r-1$ variables before x_i which forbid a values for x_i, and $s-1$ variables after x_i which forbid b values for x_i, where $a+b+c = k - \mathrm{poss}(i,x)$. (We are assuming that no value is forbidden twice, which can only increase the bound.) Let $b' = b + \mathrm{poss}(i,x)$. Then the expected cost of the description, when x_j is inserted, is

$$\frac{r}{r+s}\log b' + \frac{s}{r+s}\log(b'+c).$$

Suppose instead that x_j is split into two variables, where the first one forbids one value and the second one forbids $c-1$ values. We claim that we then get a larger bound. Indeed, we obtain

$$\frac{r(r+1)}{(r+s)(r+s+1)}\log b' + \frac{sr}{(r+s)(r+s+1)}\log(b'+1) +$$

$$+\frac{rs}{(r+s)(r+s+1)}\log(b'+c-1) + \frac{s(s+1)}{(r+s)(r+s+1)}\log(b'+c)$$

as the bound. This quantity minus the previous one is

$$\frac{rs}{(r+s)(r+s+1)}(\log(b'+1)+\log(b'+c-1)-\log b'-\log(b'+c)),$$

and indeed $(b'+1)(b'+c-1) \geq b'(b'+c)$. ∎

Now choose a solution by choosing a random permutation of the variables and assigning consistent values in that order at random.

Lemma 3 *The probability of finding a solution is at least* $\frac{1}{2\sum_i \log k_i} 2^{-\sum_i \phi_{k_i}}$*.*

Proof: The probability that the length of the encoding will not exceed one more than its expected value, bounded by $\sum_i \phi_{k_i} - \log \mathrm{value}(x)$ in Lemma 2, is at least $\frac{1}{\sum_i \log k_i}$, since encoding lengths are bounded by $\sum_i \log k_i$. The probability that x will be obtained is thus at least

$$\frac{1}{\sum_i \log k_i} 2^{-(\sum_i \phi_{k_i} - \log \mathrm{value}(x)+1)}.$$

Summing over all $x \in S$, the probability is at least

$$\begin{aligned} & \sum_{x\in S} \frac{1}{\sum_i \log k_i} 2^{-(\sum_i \phi_{k_i} - \log \mathrm{value}(x)+1)} \\ = & \sum_{x\in S} \frac{1}{2\sum_i \log k_i} 2^{-\sum_i \phi_{k_i}} \mathrm{value}(x) \\ \geq & \frac{1}{2\sum_i \log k_i} 2^{-\sum_i \phi_{k_i}} \end{aligned}$$

by an application of Lemma 1. ∎

Theorem 1 *A solution can be found in randomized time* $O^*(\prod_i \min(\frac{k_i}{2}, k_i!^{\frac{1}{k_i}}))$ *with probability approaching* 1*.*

Proof: A bound of $O^*(\prod_i k_i!^{\frac{1}{k_i}})$ follows immediately from Lemma 3, since $2^{\phi_k} = k!^{\frac{1}{k}}$. To obtain the possibly smaller factor $\frac{k}{2}$, notice that we may choose two out of k values for a variable at random, hitting the value in a chosen solution with probability $\frac{2}{k}$. Once only two values are available, say $x_1 \in \{0,1\}$, then clauses $x_1 \neq 0 \vee x_2 \neq a$ and $x_1 \neq 1 \vee x_3 \neq b$ may be replaced by $x_2 \neq a \vee x_3 \neq b$, thus eliminating x_1. ∎

3 Graph Coloring

Our approach for the $(k,2)$-satisfiability problem can be adapted to the graph k-coloring problem, leading to the algorithmic results described below. This connection should not be surprising as there is a intimate relation between the $(k_i,2)$-satisfiability and graph coloring problems. Consider a graph that is already colored with some K colors, and let k_1, k_2, ..., k_K be the sizes of the various color classes. We could, for example, create a variable for each color class i and let the variable range in value over the labels for the k_i vertices in that color class. Clearly, a satisfying solution corresponds to the choice of a single vertex from each color class. The constraints can be chosen to ensure any desirable property of the selected vertices. In fact, it is easy to see to see that constraints of the form $(x_i \neq a) \vee (x_j \neq b)$ can be used to require any of a large class of structures on the subgraph induced by the vertices selected from the color classes. For instance, these constraints can be used to rule out the selection of a vertex a in a color class i and simultaneously a vertex b in a color class j that are not adjacent in the original graph. (Of course, even more general conditions can be expressed in this form.) A specific example is to find a K-clique in the K-colorable graph; clearly, it must contain exactly one vertex from each color class. This is but one interesting problem that can be solved in the same time as the $(k_i, 2)$-satisfiability problem.

Suppose now that we wish to k-color an n-vertex graph G that is indeed k-colorable. Let

$$\phi_k = \frac{1}{k+1} \sum_{i=0}^{k-1} \left(1 + \frac{i}{\binom{k}{2}}\right) \log(k-i),$$

where the log is to base 2. Note that this ϕ_k is smaller than the one in the more general $(k,2)$-satisfiability problem. This is achieved by assuming, without loss of generality, that every vertex has at least k neighbors, giving $k+1$ placements for a vertex relative to its neighbors in the random ordering used by the algorithm. The details of the analysis are in the appendix.

Theorem 2 *A valid k-coloring, if one exists, of an n-vertex graph G can be found in randomized time $O^*(\min(\frac{k}{2}, 2^{\phi_k})^n)$ with probability approaching 1. The running time is thus $(\frac{k}{e}(1+o(1)))^n$*

The quantity 2^{ϕ_k} is asymptotically bounded by k/e. Note $2^{\phi_3} = 2^{1/3}3^{1/4} = 1.658$, $2^{\phi_4} = 2.051$, $2^{\phi_5} = 2.441$, $2^{\phi_6} = 2.828$, and $2^{\phi_7} = 3.212$. Thus $2^{\phi_k} < \frac{k}{2}$ for $k \geq 5$. It turns out that a dynamic programming algorithm can perform much better for large values of k. However, the preceding results are still interesting since the dynamic programming algorithm requires exponential space, while the space for the above algorithm is linear. Note that Eppstein [3] has presented an algorithm for the case of graph 3-coloring that runs in time $O(1.3289^n)$.

Theorem 3 *A dynamic programming algorithm can find a valid k-coloring, if one exists, of an n-vertex graph in time $O^*((1+3^{1/3})^n) = O(2.442^n)$. Within prolynomial space, for $k=3$, the running time can be improved to $O^*(3^{n/3}) = O(1.442^n)$; for $k=4$ it can be improved to $O^*(3^{n/3}3^{n/4}) = O(1.898^n)$, and for $k=5$ it can be improved to $O^*(3^{4n/5}) = O(2.408^n)$.*

Proof: The dynamic programming algorithm is based on the enumeration of *maximal* independent sets in a graph, a problem which has been studied earlier in the literature [6, 7, 12, 16, 17, 22]. The number of maximal independent sets on an n-vertex graph is at most $3^{n/3} = 1.442^n$. The proof works by induction. If a vertex has degree at least 3, then $a_n \leq a_{n-1} + a_{n-4}$ by considering whether the chosen vertex is in the maximal independent set or not. We are left with the case where all vertices have degree at most 2, i.e., a union of paths and cycles. It is easy to see that a union of triangles is the worst case.

Thus, a graph can be 3-colored in $O^*(3^{n/3})$ time, since the above proof is constructive. Similarly, it is easy to see that a graph can be 4-colored in time $O^*(3^{n/3}3^{n/4}) = O(1.898^n)$, by assuming that the first independent set has size at least $n/4$. Further, a graph can be 5-colored in $O^*(3^{n/3}(3^{0.8n/3})3^{n/5}) = O(2.408^n)$ time.

To get a k-coloring with dynamic programming, we can just consider the $\binom{n}{r}$ subsets of size r, find their maximal independent sets in time $O^*(3^{r/3})$, then determine the number of colors for a smaller set of left over vertices (already precomputed). The running time is $\sum_r \binom{n}{r} 3^{r/3} = (1+3^{1/3})^n = 2.442^n$. ■

If we want to avoid all the storage for dynamic programming, we can partition into $\lfloor k/2 \rfloor$ bipartite graphs with an additional independent set for k odd. The running time is $O^*((\lceil \frac{k}{2} \rceil)^n)$. Compare this with removing independent sets, the first one in $3^{\frac{n}{3}}$, the next one in $3^{\frac{k-1}{k}\frac{n}{3}}$, and so on up to $3^{\frac{3}{k}\frac{n}{3}}$; this gives $3^{(\frac{k+1}{6}-\frac{1}{k})n}$, which is only an improvement for $k \leq 5$ as stated in the preceding theorem.

Acknowledgements

We are indebted to the following people for several helpful discussions: Moses Charikar, Chandra Chekuri, Sudipto Guha, David Karger, Sanjeev Khanna, and Rammohan Paturi.

References

[1] E.T. Bax. Inclusion and exclusion algorithm for the Hamiltonian Path Problem. *Information Processing Letters* 47(1993): 203–207.

[2] R. Beigel and D. Eppstein. 3-Coloring in time $O(1.3446^n)$: a no-MIS algorithm. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 444-453, 1995.

[3] D. Eppstein. Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction. *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 329–337.

[4] Y. Gurevich and S. Shelah. Expected computation time for the Hamiltonian Path Problem. *SIAM Journal on Computing* 16(1987): 486–502.

[5] M. Held and R.M. Karp. A dynamic programming approach to sequencing problems. *Journal of Society of Industrial and Applied Mathematics* 10(1962): 196–210.

[6] T. Jian. An $O(2^{0.304n})$ algorithm for solving maximum independent set problem. *IEEE Transactions on Computing* C-35(1986): 847–851.

[7] D. Johnson, M. Yannakakis, and C.H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters* 27(1988): 119–123.

[8] D. Karger, R. Motwani, and M. Sudan. Approximate Graph Coloring by Semidefinite Programming. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1994.

[9] O. Kullman. Worst-case Analysis, 3-SAT Decision and Lower Bounds: Approaches to Improved SAT Algorithms. In *Proceedings of the DIMACS Workshop on Satisfiability Problem*, 1996.

[10] E.L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters* 5(1976): 66–67.

[11] B. Monien and E. Speckenmeyer. Solving Satisfiability in Less Than 2^n Steps. *Discrete Applied Mathematics* 10(1985): 287–295.

[12] J.W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics* 3(1965): 23–28.

[13] R. Motwani, K. Palem, V. Sarkar, and S. Reyen. Combining Register Allocation and Instruction Scheduling. *Technical Report STAN-CS-TN-95-22*, Department of Computer Science, Stanford University, 1995.

[14] R. Paturi, P. Pudlák, and F. Zane. Satisfiability Coding Lemma. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, 1997.

[15] R. Paturi, P. Pudlák, M.E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pp. 628–637, 1998.

[16] J.M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms* 7(1986): 425–449.

[17] M. Shindo and E. Tomita. A simple algorithm for finding a maximum clique and its worst-case time complexity. *Systems and Computers in Japan* 12(1990): 1–13.

[18] I. Schiermeyer. Solving 3-satisfiability in less than 1.579^n steps. In *Selected Papers from Workshop on Computer Science Logic*, Springer-Verlag Lecture Notes in Computer Science, volume 702, pages 379–394, 1993.

[19] I. Schiermeyer. Deciding 3-colorability in less than $O(1.415^n)$ steps. In *Proceedings of the 19th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 177–182, 1994.

[20] I. Schiermeyer. Pure Literal Look Ahead: An $O(1.497)$ 3-Satisfiability Algorithm. In *Proceedings of the Workshop on the Satisfiability Problem*, Siena, 1996.

[21] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pp. 410–414, 1999.

[22] R.E. Tarjan and A.E. Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing* 6(1977): 537–546.

[23] W. Zhang. Number of models and satisfiability of sets of clauses. *Theoretical Computer Science*, 155(1996): 277–288.

A Analysis for Graph Coloring

We now present the proof of Theorem 2 that was stated earlier. Assume that we wish to k-color an n-vertex graph G that is indeed k-colorable. Recall that

$$\phi_k = \frac{1}{k+1} \sum_{i=0}^{k-1} \left(1 + \frac{i}{\binom{k}{2}}\right) \log(k-i),$$

where the log is to base 2. For a coloring x, let

$$\mathrm{value}(x) = \prod_v \frac{1}{(k - \mathrm{nbr}(v,x))},$$

where $\mathrm{nbr}(v,x)$ is the number of colors of neighbors of vertex v in the coloring x.

Lemma 4 *Let S be the set of all valid k-colorings, and suppose S is nonempty. Then $\sum_{x \in S} \mathrm{value}(x) \geq 1$.*

Proof: Order the vertices $v_1, v_2, \ldots, v_n$. Construct a tree whose nodes at depth i (the root has depth 0) are the colorings of the first i vertices that can be extended to colorings in S. A partial coloring at depth $i-1$ has as children extensions of the coloring that assign possible colorst for v_i.

Now assign numbers to the nodes of the tree. The root has number 1. If a node has number r, and has d children, its children have number r/d. Thus the leaves have number $1/(d_1 d_2 \cdots d_n)$, where d_i is the number of children of the node at depth $i-1$ in the path from the root to the leaf. Since the number at a node is the sum of the numbers at its children, the numbers in the leaves sum to 1. For a coloring x at a leaf, we have $d_i \geq k - \mathrm{nbr}(v_i, x)$, so $\mathrm{value}(x) \geq 1/(d_1 d_2 \cdots d_n)$, and therefore $\sum_{x \in S} \mathrm{value}(x) \geq 1$. ■

Assume every vertex has degree at least k. Describe a coloring x with respect to a random permutation of the vertices, by choosing the color for each vertex out of the colors different from those of adjacent vertices preceding in the permutation.

Lemma 5 *A valid coloring x can be encoded with $\phi_k n - \log \mathrm{value}(x)$ expected number of bits.*

Proof: Assume first that x is isolated, that is the neighbors of v use $k-1$ colors for each v, so that $\mathit{value}(x) = 1$. Choose v and k of its neighbors to include all k colors. In a random permutation of these $k+1$ vertices, v falls in position $0 \leq i \leq k$, each with probability $1/(k+1)$. Only one color repeats. The probability that this color repeats before v is $\binom{i}{2}/\binom{k}{2}$. The encoding length of v then has expected value

$$\frac{1}{k+1} \sum_{i=0}^{k} \frac{\binom{i}{2}}{\binom{k}{2}} \log(k-i+1) + \left(1 - \frac{\binom{i}{2}}{\binom{k}{2}}\right) \log(k-i) = \phi_k,$$

proving the $\phi_k n$ bound.

If the neighbors of v have $k-j$ colors, we get an upper bound by replacing $\log(k-i)$ with $\log(k-i+j-1)$, that is

$$\begin{aligned} & \frac{1}{k+1}\sum_{i=0}^{k-1}\left(1+\frac{i}{\binom{k}{2}}\right)\log(k-i+j-1) \\ \leq\ & \frac{1}{k+1}\sum_{i=0}^{k-1}\left(1+\frac{i}{\binom{k}{2}}\right)(\log(k-i)+\log j) \\ =\ & \phi_k+\frac{1}{k+1}\sum_{i=0}^{k-1}\left(1+\frac{i}{\binom{k}{2}}\right)\log j \\ =\ & \phi_k+\log j=\phi_k+\log(k-\mathrm{nbr}(v,x)), \end{aligned}$$

establishing the $\phi_k n - \log \mathrm{value}(x)$ bound. ■

Now choose a coloring by picking a random permutation of the vertices and assigning valid colors in that order at random.

Lemma 6 *The probability of finding a valid coloring is at least* $\frac{1}{2(\log k)n}2^{-\phi_k n}$.

Proof: The probability that the length of the encoding will not exceed one more than its expected value, bounded by $\phi_k n - \log \mathrm{value}(x)$ in Lemma 2, is at least $\frac{1}{(\log k)n}$, since encoding lengths are bounded by $(\log k)n$. The probability that x will be obtained is thus at least $\frac{1}{(\log k)n}2^{-(\phi_k n-\log \mathrm{value}(x)+1)}$. Summing over all $x \in S$, the probability is at least

$$\begin{aligned} & \sum_{x\in S}\frac{1}{(\log k)n}2^{-(\phi_k n-\log \mathrm{value}(x)+1)} \\ =\ & \sum_{x\in S}\frac{1}{2(\log k)n}2^{-\phi_k n}\mathrm{value}(x) \\ \geq\ & \frac{1}{2(\log k)n}2^{-\phi_k n} \end{aligned}$$

by an application of Lemma 1. ■

This implies the result stated in Theorem 2; the possibly smaller $k/2$ term is obtained by selecting two colors for each vertex, at random, and solving the resulting 2-satisfiability problem.