Analysis of Random Processes via And-Or Tree Evaluation

Michael G. Luby*

Michael Mitzenmacher†

M. Amin Shokrollahi‡

TR-97-042

Abstract

We introduce a new set of probabilistic analysis tools based on the analysis of And-Or trees with random inputs. These tools provide a unifying, intuitive, and powerful framework for carrying out the analysis of several previously studied random processes, including random loss-resilient codes, solving random k-SAT formulae using the pure literal rule, the greedy algorithm for matchings in random graphs. In addition, these tools allow generalizations of these problems not previously analyzed to be analyzed in a straightforward manner. We illustrate our methodology on the three problems listed above.

*International Computer Science Institute, Berkeley, CA. Parts of this research were done while still at the Digital Equipment Corporation Systems Research Center, Palo Alto, CA. Research partially supported by NSF operating grant NCR-9416101.

†Digital Equipment Corporation, Systems Research Center, Palo Alto, CA.

‡International Computer Science Institute Berkeley, and Institut für Informatik der Universität Bonn, Germany. Research supported by a Habilitationsstipendium of the Deutsche Forschungsgemeinschaft, Grant Sh 57/1–1.

1 Introduction

We introduce a new set of probabilistic analysis tools related to the amplification method introduced by [15] and further developed and used in [17, 6]. These tools provide a unifying, intuitive, and powerful framework for carrying out the analysis of several previously studied random processes, including the random loss-resilient codes introduced in [11], the greedy algorithm for matchings in random graphs studied in [9], the threshold for solving random k-SAT formulae using the pure literal rule [5], the emergence of a giant k-core [16], and error-correcting codes introduced in [8, 12]. In addition, generalizations of these problems not previously analyzed can now be analyzed in a straightforward manner. For example, we can analyze generalizations of the loss-resilient codes considered in [11] where the goal is to recover a certain fraction of the message packets. As another example, we can analyze the behavior of the pure literal rule on random SAT formulae chosen from distributions not considered by previous analyses.

Our main tool is a simple analysis of the probability an And-Or tree formula evaluates to 1. The simple version of this And-Or tree evaluation problem is the following. Let T_ℓ be a tree of depth 2ℓ with each leaf node labeled with either 0 or 1. (The root of the tree is at depth 0, and the leaves are at depth 2ℓ.) Each node at depth $0, 2, 4, \ldots, 2\ell - 2$ is labeled as an "OR" gate (and it evaluates to the "OR" of its children), and each node at depth $1, 3, 5, \ldots, 2\ell - 1$ is labeled as an "AND" gate (and it evaluates to the "AND" of its children). We say the tree is $(d_{\mathrm{or}}, d_{\mathrm{and}})$-*regular* if each "OR" node has d_{or} children and each "AND" node has d_{and} children.

Our analysis is related to the study of amplification, initiated by Moore and Shannon [15], and continued in several works [17, 6]. Consider the probability that the root of T_ℓ evaluates to 0 when the value of each leaf is independently chosen to be 0 with probability p. Let us denote this probability as y_ℓ. One typical amplification question with respect to And-Or trees is whether or not there is a threshold phenomenon, i.e., is there a critical value ϕ such that if $p > \phi$ then y_ℓ goes to 1 as ℓ goes to infinity and if $p < \phi$ then as y_ℓ goes to 0 as ℓ goes to infinity. Of primary interest in these studies is the rate of amplification, i.e., the rate at which y_ℓ goes to either 0 or 1 as a function of ℓ.

One work that uses exactly this type of analysis is the elegant randomized construction, given in [17], of a polynomial size monotone boolean formula that computes the majority function. The basic idea behind the construction and proof exploits the fact that a (2, 2)-regular And-Or tree has a critical value of $\phi = (3-\sqrt{5})/2$, and that if $y_{\ell-1} = \phi + \epsilon$ then $y_\ell > \phi + c\epsilon$ for a constant $c > 1$. (Analogously, if $y_{\ell-1} = \phi - \epsilon$, then $y_\ell < \phi - c\epsilon$.) In further work, [4] and [6] provide beautiful proofs that the construction size of [17] is optimal, this time using amplification analysis to prove a lower bound. This type of analysis has also been noted as a possible attack for specific random graph problems, including the greedy matching algorithm of [9] and the emergence of the k-core [16].

Our work has a similar spirit to this previous work, but it differs in several ways. We generalize to allow the number of children of each node to vary in the following way. Let $(\alpha_0, \alpha_1, \ldots, \alpha_A)$ be a probability vector, i.e., $\alpha_i \geq 0$ for all $i \in \{0, \ldots, A\}$ and $\sum_{i=0}^{A} \alpha_i = 1$. Similarly, let $(\beta_0, \beta_1, \ldots, \beta_B)$ be a probability vector. Starting at the root and working down the tree, each "OR" node chooses to have i children with probability α_i independent of any other node, and similarly each "AND" node choose to have j children with probability β_j independent of any other node. (Some previous research, e.g., [6], introduced a variant of this form and used it in a limited way in their construction.) A further generalization is to allow two positive values a and b such that each "OR" node is independently short circuited to produce the value 1 with probability a, and each "AND" node is independently short circuited to produce the value 0 with probability b.

Our new analysis is based on the following simple definitions and lemma. Define

$$\begin{aligned}
\alpha(x) &= \sum_{i=0}^{A} \alpha_i \cdot x^i, \\
\beta(x) &= \sum_{i=0}^{B} \beta_i \cdot x^i, \\
f(x) &= (1-a) \cdot \alpha(1-(1-b) \cdot \beta(1-x)).
\end{aligned}$$

LEMMA 1. *Define $y_0 = p$ to be the probability a leaf node is labeled 0. Then for all $\ell \geq 1$, $y_\ell = f(y_{\ell-1})$.*

Although this lemma is simple to prove (in fact, we leave it as an exercise), as we shall see it is quite powerful.

The most significant difference between our work and previous work on amplification is our goals. For example, for the loss-resilient codes, our final goal is to design $(\alpha_0, \ldots, \alpha_A)$ and $(\beta_0, \ldots, \beta_B)$ so that the average number of children per node is not too large, the average number of children for an "OR" gate and for an "AND" gate satisfies a certain ratio, and for as small as possible a value of a and as large as possible a value of b the critical value of the tree is as close to 1 as possible.

Some of the analyses we present in the paper were previously done using different methodologies. One common technique involved modeling the random process using differential equations. This type of approach was pioneered in the analysis of algorithms domain in by Karp and Sipser, who used it to analyze a greedy algorithm for matchings in [9]. It has also been used to analyze the pure literal on random k-SAT formulae [14]. (See also [13, 14, 16] for references to other uses.) Similarly, the analysis of the loss-resilient codes described in [11] was done by modeling the random process using differential equations, solving the equations to obtain a polynomial, and using a version of Kurtz's theorem [10] to make the connection between the behavior of the random process and that of the polynomial. One of the ingredients lacking in the previous analysis of these codes was a simple intuitive connection between the polynomial solution and the original process. With the new analysis, this intuitive connection is direct and compelling. In addition, the new tools can be easily used to analyze important generalizations of the original process, which would have been much more difficult using the previous analysis. Finally, the simplicity and generality of the new analysis will undoubtably lead to a number of other applications. We do note, however, that the differential equations approach can lead to a much more sophisticated analysis of the underlying processes (see, for example, [16] or [3]). Hence we believe our analysis tools will prove most useful in conjunction with differential equations.

In the next three sections, we apply Lemma 1 to the analysis of loss-resilient codes, the pure literal rule for random k-CNF formula, and the greedy matching algorithm for random graphs.

2 Loss-Resilient Code Analysis

2.1 Essentials of the Codes

The codes described in [11] consist of a cascading sequence of random bipartite graphs. Because the code requires the same properties from all of these bipartite graphs, it is enough to consider one generic bipartite graph in the sequence when describing the encoding and decoding process and its analysis. Let G be a bipartite graph with n nodes on the left side, m nodes on the right side, and e edges in total between the nodes on the left and the right. We associate one message bit with each left node and one check bit with each right node. (This is for simplicity of description; in practice it is more efficient to associate several bytes of information with each node.) The encoding process computes the check bits from the message bits in the obvious way: the check bit associated with right node w is computed as the exclusive-or of all the message bits associated with the neighbors of w.

The entire encoding is transmitted, and we would like to recover all the message bits from a random fraction of the entire encoding, where this fraction is as small as possible. Assume inductively that all the check bits associated with the right nodes have already been recovered. Label the left nodes with a 0 if the associated message bit is missing, and with a 1 if the associated message bit has been either received directly or recovered indirectly as described below. The decoding process to recover the missing message bits invokes the following rule as long as it is applicable.

Substitution Recovery Rule: *The rule can be applied at any left node v with label 0 that has at least one right neighbor w such that all the left neighbors of w excluding v are labeled with a 1. The value of v can be recovered by computing the exclusive-or of the check bit associated with w and all the values associated with neighbors of w excluding v. Since the message bit associated with v has been recovered, the label of v is changed to 1 at this point.*

In terms of a graph process, the substitution recovery rule can be written more succinctly as follows:

Graph Substitution Recovery Rule: *A left node v with label 0 is allowed to change its label to a 1 if it has at least one right neighbor w such that all left neighbors of w except v have label 1.*

The decoding process terminates successfully with all message bits recovered if and only if the graph substitution recovery rule ends with no remaining left nodes with label 0.

2.2 New Analysis of the Original Process

The paper [11] gave an analysis of the decoding process described in the previous subsection using differential equations to model the process, and then solving these equations as polynomials. In this subsection, we obtain the same result using Lemma 1. The advantage of the analysis here is that it gives direct and intuitive insight into how the final condition arises. In the following subsections we show how this new analysis can be used to derive several additional results.

Let $(p_0, p_1, \ldots, p_L)$ and $(q_0, q_1, \ldots, q_R)$ be probability vectors. As in [11], consider choosing a random bipartite graph with n left nodes and m right nodes as follows: each node on the left is chosen to have degree i with probability p_i, and each node on the right is chosen to have degree j with probability p_j, where all choices are made independently. Counting the number e of edges using the left and the right nodes gives

$$e = n \cdot \sum_{i=0}^{L} i p_i = m \cdot \sum_{j=0}^{R} j q_j.$$

A random permutation π of $\{1, \ldots, e\}$ is chosen, and then, for all $i \in \{1, \ldots, e\}$, the edge with index i out of the left side is identified with the edge with index π_i out of the right side.

For fixed probability vectors $(p_0, p_1, \ldots, p_L)$ and $(q_0, q_1, \ldots, q_R)$ and for a fixed constant $c > 0$, we are interested in properties of such a graph as n and $m = cn$ grow to infinity.

Consider the random subgraph G_ℓ of this graph obtained by the following process: choose an edge (v, w) uniformly at random from among all edges, and then consider the subgraph G_ℓ induced by the left node v and all neighbors of v within distance 2ℓ after deleting the edge (v, w).

We claim that the probability that G_ℓ fails to be a tree is proportional to $1/n$, i.e., asymptotically this probability goes to zero as n grows to infinity for a fixed value of ℓ. Furthermore, asymptotically the distribution on the shape of G_ℓ can be described as follows. For all $i = 1, \ldots, L$, $\lambda_i := i p_i / \sum_{j=1}^{L} j p_j$ is the probability that a uniformly chosen edge is attached to a left node of degree i. Similarly, for all $i = 1, \ldots, R$, $\rho_i = i q_i / \sum_{j=1}^{R} j q_j$ is the probability that a uniformly chosen edge is attached to a right node of degree i. The asymptotic distribution on the shape of G_ℓ as n goes to infinity is as described above for a randomly chosen And-Or tree with the following parameters: the number of children of an "OR" node is $i - 1$ with probability λ_i, for all $i = 1, \ldots, L$; the number of children of an "AND" node is $i - 1$ with probability ρ_i, for all $i = 1, \ldots, R$. Hereafter, we make the assumption that G_ℓ is a tree and that the distribution on G_ℓ is the asymptotic distribution. This assumption can be shown to change the analysis by asymptotically small quantities, and these changes can be dealt with using an appropriate martingale argument, see e.g., Section 2.3.

Consider a process where at the start each left node in the graph is labeled with 0 initially with probability δ, and is labeled with 1 with probability $1 - \delta$. This corresponds to missing a random fraction δ of the message bits. The goal is to eliminate as many as possible 0 labels according to the graph substitution recovery rule described in the previous subsection, i.e., to recover as many of the missing message bits as possible using the simple decoding process.

Let us analyze the probability y_ℓ that the left node v of a uniformly chosen edge (v, w) is labeled with 0 considering the process running only on the subgraph G_ℓ induced by v. (It is clear that v will definitely change its label to 1 in the process running on the entire graph if it does so with process running just on G_ℓ.) Note that v obtains the label 1 with respect to G_1 if it is either received directly (with probability $1 - \delta$), or if for at least one of its right neighbors w' ($w \neq w'$), all left neighbors of w' excluding v are received directly. Note that v has $i-1$ right children excluding w with probability λ_i, and that for any child w' of v, w' has $i-1$ left children excluding v with probability ρ_i. Define the polynomials

$$\begin{aligned} \lambda(x) &= \sum_{i=1}^{L} \lambda_i \cdot x^{i-1}, \text{ and} \\ \rho(x) &= \sum_{i=1}^{R} \rho_i \cdot x^{i-1}. \end{aligned}$$

Then from Lemma 1, $y_\ell = \delta \cdot \lambda(1 - \rho(1 - y_{\ell-1}))$. Using this equation, the probability that v has label 0 with respect to G_ℓ can be computed iteratively starting with the equation $y_0 = \delta$. We would like that y_ℓ goes to 0 as ℓ grows. This condition will be true if

$$\delta \cdot \lambda(1 - \rho(1 - x)) < x \tag{2.1}$$

for all $x \in (0, \delta]$. (Becaue λ and ρ are continuous and the y_ℓ are decreasing, the limit of the y_ℓ is easily shown to be 0 if this condition holds.) This turns out to equivalent to the condition given in [11] for this process to end successfully.

2.3 The Overall Analysis

It is not hard to argue that the process terminates with all message values successfully recovered with high probability if the probability that a message bit is not directly received is δ and if Condition (2.1) is fulfilled. However, the details are somewhat tedious and thus we only sketch the proof here.

Suppose that λ, ρ, and δ satisfy Condition (2.1). Then for any constant $\gamma > 0$ we can set ℓ to a constant so that $y_\ell < \gamma$. If ℓ is a constant then the number of nodes in the graph G_ℓ is also a constant.

We can use this fact and standard martingale arguments to show that the true fraction of bits not recovered after ℓ rounds is highly concentrated around y_ℓ. We first utilize an edge exposure martingale to show that the number of trees of each shape is close to its expectation with high probability. With this fact, a vertex exposure martingale that tracks whether a node is labeled 1 initially or not can be used to show the strong concentration around y_ℓ. Hence the number of message bits not recovered at the end of the decoding process is greater than $\gamma' n$ with probability exponentially small in n, for $\gamma' \approx \gamma$.

Then using the expansion properties of the random graph, which follows from standard combinatorial arguments as outlined in [11], it is not hard to argue that if at most $\gamma' n$ message bits are left recovered then the decoding process fails to recover more than $O(n^{\gamma''})$ message bits with probability at most inverse polynomial in n, for some constant $\gamma'' < 1$. Finally, by a small supplement to the graph (adding a few nodes on the right and having three additional edges out of each node on the left mapping randomly to these few additional right nodes), one can see that if the process fails to recover at most $O(n^{\gamma''})$ of the message bits in the original graph, then in the supplemented graph all message bits fail to be recovered with probability at most inverse polynomial in n. From this it follows that, with high probability, when the decoding process terminates all message bits have been successfully recovered.

2.4 The Dual Inequality

In [11] a procedure is described for finding (close to) optimal right probabilities $\rho_1, \ldots, \rho_R$ for a given set of left probabilities $\lambda_1, \ldots, \lambda_L$ using a linear programming approach. However, [11] did not describe how to find the optimal left probabilities for a given set of right probabilities. Using Condition (2.1), it is easy to see how to use the methodology described in [11] to do exactly this. In fact, Condition (2.1) is in some sense the dual of the corresponding condition described in [11], which was

$$\rho(1 - \delta \cdot \lambda(1 - x)) > x \tag{2.2}$$

for all $x \in (0, 1]$. It is from Condition (2.2) that [11] shows how to find the optimal right probabilities for a given set of left probabilities. We leave it as an exercise how to use the And-Or tree analysis to easily derive Condition (2.2). It turns out that Condition (2.2) can also be derived from Condition (2.1) using a few simple algebraic manipulations. We leave this as an exercise as well.

One advantage of being able to solve for both the optimal left probabilities for a given set of right probabilities and the optimal right probabilities for a given set of left probabilities is that we can invoke a "back and forth" strategy to get a good pair of distributions. This strategy consists of starting with any given set of left and right probabilities with a given average degree, and then iteratively invoking the "find the best left for the given right" followed by "find the best right for the given left". We have tried this strategy and it gives good results, although at this point we have not proved anything about its convergence to a (possibly optimal) pair of probability distributions.

2.5 Fraction of Left Nodes Unrecovered

The new analysis of the decoding process also yields extensions that help to overcome other practical problems in the design of loss-resilient codes. In the original analysis it is assumed inductively that all the check bits are received when trying to recover the message bits. The reason we made this assumption is that in the original construction the cascading sequence of bipartite graphs is completed by adding a standard loss-resilient code at the last level.

There are some practical problems with this. One annoyance is that it is inconvenient to combine two different types of codes. A more serious problem is that standard loss-resilient codes take quadratic time to encode and decode. Suppose the message is stretched to an encoding twice its length. In order to have the combined code run in linear time, this implies that the last graph in the cascading sequence has $\sqrt{n}$ left nodes, where n is the number of nodes associated with the original message, i.e., there are $\log(n)/2$ graphs in the sequence. In the analysis, we assume that an equal fraction of the nodes in each level of the graph are received. However, there is variance in this fraction at each level, with the worst expected fractional variance at the last level of $1/\sqrt[4]{n}$. Thus, if a message of length $65,536$ is stretched to an encoding of length $131,072$, then just because of the variance of $1/\sqrt[4]{n} = 0.063$, we expect to have to receive 1.063 times the message length of the encoding in order to recover the message.

A solution to this problem is to use many fewer levels of graphs in the cascade, and at the last level also use a random graph in place of a standard loss resilient code. We have tried this idea, with the last graph chosen from an appropriate distribution, and it works quite well. For example, using only three levels of graphs we can reliably recover a message of length $65,536$ from a random portion of length $67,700$ (i.e., 1.033 times the optimal of $65,536$) of an encoding of length $131,072$.

To design the graph for this solution, we need to analyze the decoding process when a random portion of both the message bits and the check bits are missing. With the And-Or tree analysis, this is straightforward. Recall the terminology established in Subsection 2.2. Suppose that a random fraction δ of the message bits are not received directly and a random fraction δ' of the check bits are not received directly. The generalization of Condition (2.1) for this case when there are losses on both sides is that the process terminates in a state where a uniformly chosen edge is adjacent to a left node with a missing message bit with probability at most γ if

$$\delta \cdot \lambda(1 - (1 - \delta') \cdot \rho(1 - x)) < x \tag{2.3}$$

for all $x \in (\gamma, \delta]$.

The more general version of Condition (2.2), when a fraction δ' of the right nodes are missing, is

$$\rho(1 - \delta \cdot \lambda(1 - (1 - \delta')x)) > x \tag{2.4}$$

The Condition (2.4) is not possible to satisfy for all $x \in (0, 1]$ if $\delta' > 0$, for any value of δ. This is because there is a constant probability that all the right neighbors of a missing left node are also missing, e.g., if the left node has degree d then the probability is δ'^d. However, it turns out to be an interesting question to see what fraction of the left nodes can be recovered when a fraction δ' of the right nodes are missing. The answer to this question can be used to design cascading codes where the decoding process moves from right to left bootstrapping up to recover a higher and higher fraction of nodes at each successive decoded layer of the graph until it is in practice able to recover all of the first (message) layer. That is, the constant fraction left unrecovered is so small that in practice all nodes corresponding to the message are recovered.

Given the fractions of left nodes p_i and right nodes q_i of degree i for all i, $\lambda(x)$ and $\rho(x)$ can be easily derived, and then the largest value x^* for which Condition (2.4) is valid can be computed. We show here how to compute the fraction of unrecovered nodes on the left at this final value x^*.

The value of x^* has a natural interpretation, i.e., it is the fraction of edges (v, w) for which all of the left neighbors of w, excluding v, have label 1 at the end of the process. Thus, this is the fraction of edges (v, w) which could cause v to receive the label 1, assuming that w is directly received. Define

$$p(x) = \sum_{i=0}^{L} p_i \cdot x^i.$$

From this interpretation, it can be seen that the fraction of unrecovered left nodes at the termination of the process is

$$\delta \cdot p(1 - (1 - \delta')x^*).$$

This is because $y = 1 - (1 - \delta')x^*$ is the fraction of edges (v, w) which cannot help to recover v. Thus, a left node v of degree i is not recovered at the end with probability δ (its original missing probability) times y^i, and there is a p_i fraction of such left nodes.

3 Pure Literal Analysis

In this section, we consider a simple heuristic, called the *pure literal rule*, for finding a satisfying truth assignment to a boolean formula. The behavior of the pure literal rule has been studied previously with respect to randomly chosen k-SAT formulae ([5, 14]). (See also [7] for related results using more sophisticated heuristics.) Here, we show how the tree analysis gives a direct explanation of the behavior of the pure literal rule for a randomly chosen k-SAT formula with respect to the same distributions considered in [5] and [14]. With this new analysis, it is also straightforward to analyze distributions that were not previously considered and which would be much harder to analyze using previous techniques applied to this problem.

A k-SAT formula F with m clauses on n variables $\{X_1, \ldots, X_n\}$ consists of m clauses $C_1, \ldots, C_m$, each clause containing exactly k of the $2n$ possible literals

$$\Lambda := \{X_1, \bar{X}_1, \ldots, X_n, \bar{X}_n\}.$$

Then the formula F is the "AND" of the m clauses, and each clause is the "OR" of the k literals it contains, i.e., for any 0/1 assignment to the variables, F evaluates to 1 if and only if in each clause there is at least one literal that has value 1 with respect to the assignment. The most widely studied distribution on F is the *uniform* distribution. For fixed value of k, m, n the uniform distribution on choosing a formula F can be described as follows: for each $i \in \{1, \ldots, m\}$ and for each $j \in \{1, \ldots, k\}$, each of the $2n$ possible literals is chosen with equal probability to be the jth literal in clause C_i.

The *pure literal rule* heuristic for finding a satisfying assignment consists of repeated application of the following:

Pure Literal Rule: *While there is a literal $Z \in \Lambda$ that appears in zero clauses, remove all clauses containing the negation $\bar{Z}$ of Z, assign $\bar{Z}$ the value 1 (and thus Z is assigned 0), and remove both Z and $\bar{Z}$ from Λ.*

The problem of interest is to study the asymptotic behavior of the pure literal rule with respect to uniformly chosen k-SAT formulae for a fixed value of k, and for a fixed ratio $c = m/n$ of the number of clauses to the number of variables, as the number of variables n grows to infinity. The more particular question is for which values of k and c will the pure literal rule almost surely (with respect to a uniformly chosen formula F) find an assignment which makes F evaluate to 1 as n goes to infinity.

Similar to the loss-resilient codes, we can describe the structure of the formula F as a bipartite graph, only in this case the edges are labeled. There are n right nodes in the graph corresponding to the variables, and there are m left nodes corresponding to the clauses. There is an edge labeled "+" from variable X to all clauses that contain X, and there is an edge labeled "−" from variable X to all clauses that contain $\bar{X}$.

One can describe the behavior of the pure literal rule on this graph. The pure literal rule is equivalent to repeated application of the following process on this graph:

Graph Pure Literal Rule: *If there is a variable X such that all edges touching X have the same label (either all "+" or all "−") then delete X, all neighboring clauses of X, and all edges touching any of these nodes.*

The pure literal rule finds an assignment that satisfies the formula if and only if there are no remaining right nodes after all possible applications of this process have been made.

We describe a general way of choosing a random formula F in the terminology of bipartite graphs. Let $(p_0, p_1, \ldots, p_L)$ and $(q_0, q_1, \ldots, q_R)$ be probability vectors. Suppose each clause chooses independently to have degree j with probability p_j. Suppose each variable X chooses independently to have i edges attached to it with the same label with probability q_i, and the distribution is the same for both possible labels. Counting the number e of edges using the left and the right nodes gives

$$e = m \cdot \sum_{j=0}^{R} j p_j = 2n \cdot \sum_{i=0}^{L} i q_i.$$

A random permutation π of $\{1, \ldots, e\}$ is chosen, and then, for all $i \in \{1, \ldots, e\}$, the edge with index i out of the left side is identified with the edge with index π_i out of the right side.

For the special case of the uniform distribution on k-SAT, each clause has degree k, and the number of edges with the same label out of each variable (corresponding to the number of appearances of the corresponding literal in clauses) is asymptotically distributed according to the Poisson distribution with mean $\theta = km/2n$ as n goes to infinity, i.e., the probability that a particular literal appears in i clauses is asymptotically $\exp(\theta) \cdot \theta^i/\theta!$.

Consider the random subgraph G_ℓ of this graph obtained as follows: choose an edge uniformly at random from among all edges, and suppose it is an edge between clause C and variable X with label $* \in \{+, -\}$. Consider the subgraph G_ℓ obtained by the following search. Consider variable X to be at the zeroth level of the search. Follow all the edges out of X with the opposite label of $*$. This leads to a first level of clause nodes. Let C' be one of the clauses at the first level. Follow all edges out of C' except the edge that led into C'. This leads to a second level of variable nodes. Let $*' \in \{+,-\}$ be the label of an edge from C' to some variable X'. Follow all the edges out of X' with the opposite label of $*'$. In a similar pattern, continue this breadth first search out to level 2ℓ.

As was the case for the loss-resilient codes, G_ℓ is a tree with high probability for a fixed value of ℓ as n goes to infinity. Furthermore, asymptotically the distribution on the shape of G_ℓ can be described as follows. For all $i = 1, \ldots, L$, let $\lambda_i = ip_i / \sum_{j=1}^{L} jp_j$ be the probability that a uniformly chosen edge is attached to a clause node of degree i. For all $i = 0, \ldots, R$, let $\rho_i = q_i$ be the probability that a uniformly chosen edge is attached to a variable node with i edges of the opposite label attached. Then the distribution on the shape of G_ℓ is as described above for a randomly chosen And-Or tree with the following parameters: the number of children of a clause node is $i-1$ with probability λ_i, for all $i = 1, \ldots, L$. The number of children of a variable node is i with probability ρ_i, for all $i = 0, \ldots, R$.

Consider the following labeling process of the nodes in the tree G_ℓ that starts at the leaves at level 2ℓ and works up towards the root at level 0. A leaf variable node is labeled with a 1 if and only if it would have no descendants if the tree were extended one additional level. An internal variable node is labeled with a 1 if and only if it either has no direct descendants or else they are all labeled with a 1. A clause node is labeled with a 1 if and only if at least one direct descendant is labeled with a 1. It can be checked that if the root node receives the label 1 then the pure literal rule will give that variable an assignment.

Let y_ℓ be the probability that the root node of G_ℓ will receive the label 1 by the above labeling process. Define the polynomials

$$\begin{aligned} \lambda(x) &= \sum_{i=1}^{L} \lambda_i \cdot x^{i-1} \text{ and} \\ \rho(x) &= \sum_{i=0}^{R} \rho_i \cdot x^{i}. \end{aligned}$$

From Lemma 1 it follows that this can be expressed as

$$y_\ell = \rho(1 - \lambda(1 - y_{\ell-1}))$$

In order for the pure literal rule to end with a complete assignment that satisfies the formula, we want all variables to disappear from the formula, or equivalently, receive the label 1. This means that we want

$$\rho(1 - \lambda(1 - y)) > y \tag{3.5}$$

for all y in the range $[\rho_0, 1)$ (Note that $y_0 = \rho_0$.)

For a uniformly chosen k-SAT formula we have $\lambda(x) = x^{k-1}$ and $\rho(x) = \exp(\theta(x-1))$, where $\theta = kc/2$. For a specific k we can easily determine the threshold value c for which (3.5) is satisfied. In particular, for $k = 3$ we obtain the value $c \sim 1.63$. This result has been found previously by [5] and [14] using a different approach. The advantage of the tree analysis approach employed in this paper is that, with little additional difficulty, it is easily possible to analyze substantially different distributions for choosing the formula, merely by establishing the proper functions $\rho(x)$ and $\lambda(x)$.

It requires some additional technical work, as in Section 2.3, to prove that if (3.5) is satisfied then the pure literal rule finds a solution with high probability. Specifically, we need an expansion based argument like that given in [5, Lemma 4.4] to show that once all but a constant fraction of the literals are assigned values, the process must complete with high probability. Also, technically we have only proven one direction, namely that the pure literal rule finds a solution is (3.5) is satisfied. In fact we can also show that if (3.5) is not satisfied, that is if c is chosen larger than the threshold, then with high probability the pure literal rule does not find a solution. Unfortunately, doing so appears to require using the differential equations based approach, as our tree-based approach only shows that there is a witness for the satisfiability of a random formula with high probability. (See, for example, the discussion in [16], or the proof of [9, Theorem 9].) Intuitively, however, it is clear that as long as the limiting behavior of y_ℓ behaves properly, this methodology finds the correct threshold.

4 Greedy Matching Analysis

In the paper [9], the analysis of a simple and fast heuristic for finding matchings was described and analyzed with respect to randomly chosen graphs. (This analysis has since been extended in [3].) The tree approach provides an analysis of what they call "Phase 1"; in fact, Karp and Sipser provide an argument based on a similar tree argument. The details are similar to, but somewhat different than, those presented above to analyze the loss-resilient codes. As mentioned above, the advantage of the tree analysis is that it can be adopted to analyze a variety of distributions on the graph with little additional effort.

The first phase of [9] is a greedy matching algorithm in a random graph. The basic step of the first phase of their algorithm is the following.

Greedy Matching Step: *Find a node v of degree one in the graph; match it to its unique neighbor w; remove v, w, and all edges touching either v or w, from the graph.*

This matching step is applied iteratively until there are no degree one nodes in the graph. The basic quantity of interest is the expected number of edges in the matching produced by this phase in a random graph.

In the work of [9], this basic quantity is analyzed with respect to a random graph with n nodes chosen as follows: each edge is chosen to exist with probability $\theta/(n-1)$ independently of all other edges. Using the tree approach, we can easily generalize this to the following. Let $(p_0, p_1, \ldots, p_L)$ be a probability vector. Each node is chosen to have degree i with probability p_i. The distribution analyzed by [9] is the special case where $p_i = \exp(-\theta)(\theta)^i/i!$.

Once the graph is fixed, the edges in the matching produced by repeated application of the greedy matching step described above depend on the order in which the nodes are chosen. Nevertheless, it is clear that the size of the matching produced by repeated application of the greedy matching step does not depend on this ordering. To be able to analyze the size of the matching, we now describe an order invariant labeling process on the graph which will be used to analyze the matching process.

For each edge (v, w) in the graph, we form directed edge $\langle v, w\rangle$ pointing from v to w and directed edge $\langle w, v\rangle$ pointing from w to v. The process labels each directed edge with one of the three symbols $\{?, 0, 1\}$. Initially, each directed edge is labeled with ?. The labeling process consists of repeated application of either one of the two cases in the greedy labeling step until no more applications are possible.

Greedy Labeling Step:

- *Label directed edge $\langle v, w\rangle$ with a 1 if all (possibly zero) directed edges pointing out of w excluding directed edge $\langle w, v\rangle$ are labeled 0.*

- *Label directed edge $\langle v, w\rangle$ with a 0 if at least one directed edge pointing out of w excluding directed edge $\langle w, v\rangle$ is labeled 1.*

The key point is that there is the following correspondence between directed edges with label 1 and the matching produced by any execution of the greedy matching process.

- If directed edge $\langle v, w\rangle$ is labeled 1 and the reverse directed edge $\langle w, v\rangle$ is also labeled 1 then edge (v, w) is in the matching for every execution of the greedy matching process.

- For all $k \geq 1$, if there are k directed edges labeled 1 pointing out of a node then in every execution of the greedy matching process exactly one of these k edges (considered as undirected edges) is in the matching.

- For every execution of the greedy matching process, for each edge (v, w) in the matching produced by the execution, at least one the directed edges $\langle v, w\rangle$ and $\langle w, v\rangle$ is labeled 1.

From this, we can calculate the size of the matching produced by any execution of the greedy matching process as follows. Let M be the set of directed edges labeled 1 for which the reverse directed edge is also labeled 1. For all $k \geq 2$, let N_k be the set of directed edges labeled 1 that are pointing out of a node w such that w has in total k directed edges labeled 1 pointing out of it. Then the size of the matching is exactly

$$\sum_{i=1}^{L} |N_k|/k - |M|/2. \tag{4.6}$$

We first justify the $\sum_{i=2}^{L} |N_k|/k$ terms in (4.6). Consider a node v with $k \geq 2$ directed edges with label 1 pointing out of it. Note that there cannot be any directed edges with label 1 pointing into v. Thus the directed edges in M and N_k for $k \geq 2$ are disjoint. Furthermore, the k directed edges labeled 1 pointing out of v are all in N_k, and there is exactly one corresponding matched edge in any execution of the greedy matching process. These k edges contribute $k/k = 1$ as required to these terms.

We now justify the $|N_1| - |M|/2$ terms in (4.6). Consider a node v with exactly one directed edge $\langle v, w \rangle$ with label 1 pointing out of it. Then the edge (v, w) is in the matching produced by the greedy matching process, and the total contribution to the size of the matching should be 1. There are two cases to consider, depending on whether directed edge $\langle w, v \rangle$ is labeled 1 or not. Suppose $\langle w, v \rangle$ is not labeled 1. Then neither $\langle v, w \rangle$ nor $\langle w, v \rangle$ is in M, and only $\langle v, w \rangle$ is in N_1. These two directed edges contribute $1 - 0/2 = 1$ to (4.6), as required. Suppose $\langle w, v \rangle$ is labeled 1. Then both $\langle v, w \rangle$ and $\langle w, v \rangle$ are in both M and N_1. These two directed edges contribute $2 - 2/2 = 1$ to (4.6), as required.

The strategy now is to first estimate the probability that a uniformly chosen directed edge is labeled 1 by the greedy labeling process, and then to use this probability to estimate the size of the matching produced by the greedy matching algorithm based on the observations just made. As in the analysis of loss-resilient codes, we are interested in properties of such a graph as n grows to infinity. Also as before, it turns out that it is easiest to analyze the properties by considering a randomly chosen edge from the graph. Fix a directed edge $\langle v, w \rangle$ in the graph, and let $G_\ell \langle v, w \rangle$ be the subgraph induced by the edge $\langle v, w \rangle$ and all neighbors of w reachable within distance 2ℓ, following the rule that if node w' is reached using directed edge $\langle v', w' \rangle$ then the reverse edge $\langle w', v' \rangle$ is not used. As was true for the loss-resilient code analysis, the probability that $G_\ell \langle v, w \rangle$ fails to be a directed tree for a uniformly chosen edge $\langle v, w \rangle$ of the graph is proportional to $1/n$, i.e., asymptotically this probability goes to zero as n grows to infinity for a fixed value of ℓ. Furthermore, asymptotically the distribution on the shape of $G_\ell \langle v, w \rangle$ for uniformly chosen $\langle v, w \rangle$ can be described as follows. For all $i = 1, \ldots, L$, $\lambda_i := ip_i / \sum_{j=1}^{L} jp_j$ is the probability that a uniformly chosen edge is attached to a node of degree i. Each node in the tree $G_\ell \langle v, w \rangle$ at distance less than 2ℓ from the root has $i - 1$ children with probability λ_i, for all $i = 1, \ldots, L$.

Consider the following labeling process on the directed edges of tree $G_\ell \langle v, w \rangle$. The tree labeling process starts with all nodes labeled ?, and then works up from the leaves towards the root. Each directed edge $\langle v', w' \rangle$ pointing to a leaf node w' at distance 2ℓ from w is labeled with a 1 if there is no directed edge other than the reverse edge $\langle w', v' \rangle$ pointing out of w' in the entire graph. The labeling of the rest of the tree, and the directed edge $\langle v, w \rangle$, works according to the greedy labeling step described previously. The difference between the tree labeling and the greedy labeling process is that the tree labeling makes a decision to label a leaf with a 1 only in the case when the leaf has no other edges attached to it, whereas in the greedy labeling process the leaf may also receive the label 1 when the leaf has degree greater than 1 in the original graph.

It is not hard to verify that if the directed edge $\langle v, w\rangle$ receives the label 1 in the tree labeling of $G_\ell\langle v, w\rangle$ then it receives the label 1 in the greedy labeling process. Although the reverse is not true, the claim is that as ℓ grows, the probability of these two events approach one another. More formally, let z be the probability that a uniformly chosen directed edge receives the label 1 in the greedy labeling process on a randomly chosen graph. Let y_ℓ be the probability that a uniformly chosen directed edge $\langle v, w\rangle$ receives the label 1 in the tree labeling of $G_\ell\langle v, w\rangle$. It is clear that $z \geq y_\ell$ for all ℓ. We claim that y_ℓ approaches z as ℓ grows; however, the only justification we currently know for this relies on the differential equation approach. (See Theorem 9 of [9].) We assume that this is the case hereafter.

Let us analyze the probability y_ℓ that a uniformly chosen directed edge $\langle v, w\rangle$ is labeled with 1 in the tree labeling of $G_\ell\langle v, w\rangle$. As before, define the polynomial

$$\lambda(x) \;=\; \sum_{i=1}^{L} \lambda_i \cdot x^{i-1}.$$

Then from Lemma 1 and the description of the process, $y_\ell = \lambda(1-\lambda(1-y_{\ell-1}))$. Using this equation, the y_ℓ can be computed iteratively starting with the equation $y_0 = \lambda_1$.

Let y be the asymptotic limit of y_ℓ as ℓ grows. We need a couple of additional observations in order to be able to compute the expected size of the matching returned by the greedy matching process as a function of y. The observations are that asymptotically, as n goes to infinity:

- The labels received by a given set of directed edges pointing out of a particular node are independent of one another.
- The labels of a given directed edge and its reverse directed edge are independent of one another.

From these observations, we can conclude that the probability a randomly chosen directed edge is in N_k is

$$\alpha_k := \sum_{i=k}^{L} \lambda_i y^k (1-y)^{i-k} \binom{i-1}{k-1}.$$

Furthermore, the probability a randomly chosen directed edge is in M is y^2. Thus, the expected size of the matching produced by the greedy algorithm is equal to the expected number of directed edges in the graph times

$$\sum_{k=1}^{L} \alpha_k/k - y^2/2.$$

5 Acknowledgements

The second author thanks Alan Frieze for the references to [16] and [3], and thanks Alan Frieze and Andrei Broder for several useful discussions.

References

[1] N. Alon, J. Edmonds, M. Luby, "Linear Time Erasure Codes With Nearly Optimal Recovery", *Proc. of the* 36^{th} *Annual Symp. on Foundations of Computer Science*, 1995, pp. 512-519.

[2] N. Alon, M. Luby, "A Linear Time Erasure-Resilient Code With Nearly Optimal Recovery", *IEEE Transactions on Information Theory* (special issue devoted to coding theory), Vol. 42, No. 6, November 1996, pp. 1732–1736.

[3] J. Aronson, A. Frieze, B. G. Pittel, "Maximum Matchings in Sparse Random Graphs: Karp-Sipser revisited", preprint.

[4] R. Boppana, "Amplification of probabilistic Boolean formulas", *Advances in Computer Research*, Vol. 5: Randomness and Computation, JAI Press, Greenwich, CI, 1989, pp. 27–45.

[5] A. Broder, A. Frieze, E. Upfal, "On the Satisfiability and Maximum Satisfiability of Random 3-CNF Formulas", *Proc. of the* 4^{th} *ACM-SIAM Symp. on Discrete Algorithms*, 1993, pp. 322-330.

[6] M. Dubiner, U. Zwick, "Amplification by Read-Once Formulas", *Siam J. on Computing*, Vol. 26, No. 1, Feb. 1997, pp. 15–38.

[7] A. Frieze, S. Suen, "Analysis of Two Simple Heuristics on a Random Instance of k-SAT", *J. of Algorithms*, Vol. 20, 1996, pp. 312–355.

[8] R. G. Gallager, **Low-Density Parity-Check Codes**, MIT Press, 1963.

[9] R. Karp, M. Sipser, "Maximum Matchings in Sparse Random Graphs", *FOCS*, 1981, pp. 364–375.

[10] T.G. Kurtz, **Approximation of Population Processes**, CBMS-NSF Regional Conf. Series in Applied Math, SIAM, 1981.

[11] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. Spielman, V. Stemann, "Practical Loss-Resilient Codes", *Proc.* 29^{th} *Symp. on Theory of Computing*, 1997, pp. 150–159.

[12] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. Spielman, "Analysis of Low Density Codes and Improved Designs using Irregular Graphs", submitted to STOC 1998.

[13] M. Mitzenmacher, Ph.D. thesis. University of California, Berkeley, 1996.

[14] M. Mitzenmacher, "Tight Thresholds for the Pure Literal Rule", *DEC/SRC Technical Note 1997-011*, June 1997.

[15] E. Moore, C. Shannon, "Reliable Circuits Using Less Reliable Relays", *J. Franklin Inst.*, 262, 1956, pp. 191–208 and 281–297.

[16] B. Pittel, J. Spencer, N. Wormald, "Sudden Emergence of a Giant k-Core in a Random Graph", *J. of Combinatorial Theory, Series B*, 67, 1996, pp. 111–151.

[17] L. G. Valiant, "Short Monotone Formulae for the Majority Function", *J. of Algorithms*, Vol. 5, 1984, pp. 363–366.