An analysis of the size of the optimal dominating sets in random recursive trees, using the Cockayne-Goodman-Hedetniemi algorithm *

Colin Cooper[1] Michele Zito[2]

[1]Department of Computer Science, King's College
London WC2R 2LS (UK). e-mail: colin.cooper@kcl.ac.uk

[2]Department of Computer Science, University of Liverpool
Ashton Street, Liverpool L69 3BX (UK). e-mail: michele@liverpool.ac.uk

April 23, 2007

Abstract

A random recursive tree on n vertices is either a single isolated vertex (for $n = 1$) or is a vertex v_n connected to a vertex chosen uniformly at random from a random recursive tree on $n - 1$ vertices. Such trees have been studied before (see [11]) as models of boolean circuits. More recently, modifications of such models [2], have been used to model for the web and other "power-law" networks.

A smallest dominating set in a tree can be found in linear time using the algorithm of Cockayne, Goodman and Hedetniemi [4].

We prove that there exists a constant $d \simeq 0.3745...$ such that the size of a smallest dominating set in a random recursive tree on n vertices is $dn + o(n)$ with probability approaching one as n tends to infinity. The result is obtained by analysing the algorithm of Cockayne, Goodman and Hedetniemi.

*This research was supported by EPSRC grant EP/DO59372/1. The first author was also supported in this work by a LaBRI ENSIERB visiting research fellowship.

1 Introduction

A vertex in a graph *dominates* all vertices that are adjacent to it. A *dominating set* for a graph $G = (V, E)$ is a set $\mathcal{D} \subseteq V$ such that each vertex in $V \setminus \mathcal{D}$ is dominated by some vertex in $\mathcal{D}$. Let $\gamma = \gamma(G)$ denote the size of the smallest dominating sets in G. The *minimum dominating set* problem (MDS) asks for a dominating set of size γ.

Dominating sets play an important role in many practical applications, e.g. in the context of distributed computing or mobile ad-hoc networks [1, 5, 12]. The reader is referred to [8, 9] for an in-depth view of the subject. The typical fundamental task in such applications is to select a subset of nodes in the network that will 'provide' a certain service to all other vertices. For this to be time-efficient, all other vertices must be directly connected to the selected nodes, and in order for it to be cost-effective, the number of selected nodes must be minimal.

The MDS problem is NP-hard [7] and, moreover, it is not likely that it may be approximated effectively [6]. Polynomial time algorithms exist on special classes of graphs. In particular it is solvable in linear time on trees. This was shown by Cockayne, Goodman and Hedetniemi in [4], where they present an algorithm (the CGH Algorithm) to compute an optimal dominating set of a tree.

The problem has also been studied in general classes of random graphs. In the $G(n, p)$ model where n distinct vertices are given and each of the $\binom{n}{2}$ possible edges is chosen independently with probability p (see [3]) the value of γ can be pin-pointed quite precisely, provided p is not too small compared to n [13]. In random regular graphs of degree r (see for example results in the *configuration model* [14] and references therein) upper and lower bounds are known.

In this paper we provide a tight analysis of the CGH algorithm under the assumption that the input is a random recursive tree. Our analysis shows that, although different trees can have optimal dominating sets of different sizes, the smallest dominating sets of the vast majority of trees on n vertices, when n tends to infinity, contain about $0.3745n$ vertices.

2 Algorithm, Model and Main Result

The Cockayne, Goodman and Hedetniemi (CGH) Algorithm. The input to the algorithm is a tree on n vertices. The vertices of such tree are labelled either B (for "bound") or R (for "required") or F (for "free"). All vertices labelled R end up in the dominating set. If v is a vertex in T then $\ell(v)$ is its label. Initially all vertices are labelled B. The process is described by the following pseudo-code.

```
Input: A tree T on n nodes (initially labelled B);
Output: A dominating set D of T.

for n − 1 times
    pick a leaf v;
    if ℓ(v) = B
        ℓ(parent(v)) ← R;
    else if ℓ(v) = R
        if ℓ(parent(v)) ≠ R
            ℓ(parent(v)) ← F;
        add v to D;
    remove v from the tree;
if ℓ(v) ≠ F
    add v to D;
```

We refer to a *step* as one iteration of the main loop of the algorithm. The *final step* is the execution of the last if-statement in the code above. For convenience the steps are labeled downward from step n, the first step, to step 1, the final step. During each step, the algorithm removes one vertex of T. We denote by $T(t)$ the (sub)tree on t vertices resulting from the execution of the first $n-t$ steps of the algorithm. Similarly $B(t)$ (resp. $R(t)$, $F(t)$) denotes the set of bound (resp. required and free) vertices at step t.

Before each step of the process the sets $B(t)$, $R(t)$, and $F(t)$ partition the vertex set of $T(t)$ (so that, in particular $|B(t)|+|R(t)|+|F(t)| = t$). At the start of step t, an arbitrary leaf v_t is deleted from $T(t)$, so that the current tree $T(t-1)$ becomes $T(t) - v_t$. The sets $B(t-1)$, $R(t-1)$, $F(j-1)$, of the tree $T(t-1)$ are then updated as follows:

If $\ell(v_t) =$ X then $X(t-1)$ is defined as $X(t) - v$.

Let u be the neighbour of v in $T(t)$

If $v \in B(t)$ and $u \in B(t) \cup F(t)$ then define $R(t-1)$ as $R(t) \cup \{u\}$.

If $v \in R(t)$ and $u \in B(t)$ then define $F(t-1)$ as $F(t) \cup \{u\}$.

At the end of the algorithm, $\mathcal{D}$ (which is filled by the CGH algorithm with the required vertices) contains the vertices of a smallest dominating set of the tree (for the correctness proof of the process the reader is referred to [4]).

It is an important and remarkable fact of the CGH Algorithm, that the order in which the leaves are processed does not affect the size of the resulting dominating set.

Random recursive trees. A random recursive tree on n vertices is either a single isolated vertex v_1 (for $n = 1$) or (for $n \geq 2$) a vertex v_n connected to a vertex u_n chosen uniformly at random

(uar) from a random recursive tree on $n-1$ vertices. This process defines a probability space $\boldsymbol{T}_n$ on labelled trees on n vertices with uniform measure $1/(n-1)!$.

The sequence $(v_1, v_2, ...v_n)$ records the order in which the vertices are added to the tree, and $(u_2, .., u_n)$ their neighbour vertex u_j in the tree. We preserve this notation throughout the paper. The vertex v_1 is the *root* of the tree.

We apply the CGH Algorithm to the vertices $v_n, v_{n-1}, ..., v_1$ in that order. Thus at step $t = n, ..., 2$, v_t is a leaf with neighbour u_t chosen uar from $v_{t-1}, ..., v_1$.

Theorem 1 *For $T \in \boldsymbol{T}_n$ let D_T denote the size of the dominating set returned by running CGH Algorithm on T, and let $\mathbf{E}D(n)$ denote the expected size of such set. Then*

(i) $\mathbf{Pr}(|D_T - \mathbf{E}D(n))| \geq 2\sqrt{Kn\log n}) = O(n^{-K})$,

(ii) *Let $d(n) = \mathbf{E}D(n)/n$, then for $n \geq 5 \times 10^8$, $d(n) = 0.3745...$*

The choice of $n \geq 5 \times 10^8$, is of course arbitrary, and reflects the imprecision of our estimates of the relevant random variables, and the crudity of our numerical estimation procedures.

As a consequence of Theorem 1 and the analysis in [4] we have the following

Corollary 1 *$\gamma(T) \simeq 0.3745n$ with probability approaching one as n tends to infinity if $T \in \boldsymbol{T}_n$.*

3 Proof of Theorem 1

Distribution of vertex labels. We prove in the next lemma, that if we remove the vertices from the tree in the order $v_n, v_{n-1}, ..., v_1$ then at the start of step $t = n, ..., 1$ the partition $(B(t), R(t), F(t))$ is equally likely to be any partition of $[t]$ ($[t]$ denotes the set $\{1, 2, ..., t\}$) with sets of the given sizes.

This means that

$$\mathbf{Pr}(v_t \in X(t)) = \frac{|X(t)|}{t} \qquad X = B, R, F,$$

which will allow us to write down a recurrence for the expected values of (the sizes of) these sets, and hence arrive at an estimate for the expected size of the resulting dominating set.

Lemma 1 *Given $|B(t)| = b, |R(t)| = r, |F(t)| = f$, $(B(t), R(t), F(t))$ is a uar partition of $[t]$, and thus*

$$\mathbf{Pr}(B(t), R(t), F(t)) = \frac{1}{\binom{t}{b\ r\ f}}.$$

Proof. This is certainly true at the start, and u_n is a uar vertex of $[n-1]$ by the construction of the tree process, so that $(B(t), R(t), F(t))$ is a random $(n-2, 1, 0)$ partition. Without loss of generality, for general t consider the case

$$(R(t), F(t), B(t)) \to (R(t-1), F(t-1), B(t-1))$$

given by

$$(r, f, b) \to (r-1, f+1, b-1).$$

For this to occur the currently removed leaf v_t must be in $R(t)$ and v_t chose $u_t \in B(t)$ as its out-neighbour.

Let X be the set of triples $\{x = (R(t), F(t), B(t))\}$ and let Y be the set $\{y = (R(t-1), F(t-1), B(t-1))\}$. Consider the bipartite graph $G = (X, Y, E)$ where there is an edge $e = (x, y)$ in E, if $y = (R(t) - v_t, F(t) + u_t, B(t) - u_t)$.

Let $|X| = n, |Y| = m$. The out-degree of any vertex x is b, as any $u_t \in B(t)$ can be chosen. The in-degree of any vertex y is $f+1$ as any of the $|F(t-1)| = f+1$ vertices could be the reallocated u_t.

As in-degree equals out-degree in G, we have $nb = m(f+1)$. Under the uar hypothesis $n = \binom{t-1}{r-1,f,b}$, as $v_t \in R(t)$ is given. Thus $m = \binom{t-1}{r-1,f+1,b-1}$, which is the correct size.

Expressing this as probabilities, the uar hypothesis conditioned on $v_t \in R(t)$ gives any $x \in X$ is equally likely; so $\mathbf{Pr}(x) = 1/n$. Then u_t is uar in $\{v_1, ..., v_{t-1}\}$ and conditioning on $u_t \in B(t)$, the probability x obtains a particular $y \in Y$ is $1/b$. There are $f+1$ distinct X elements which give rise to a given y, so

$$\mathbf{Pr}(y) = (f+1)\frac{1}{n}\frac{1}{b} = \frac{1}{m},$$

so any $y \in Y$ is equally likely (uar hypothesis). □

Recurrence for $\mathbf{E}B(t), \mathbf{E}R(t)$. We next turn our attention to the expected values of $|B(t)|, |R(t)|$. To simplify notation we use X to denote both the set X and its size $|X|$ whenever the context is clear.

Given $B(t)$,and $R(t)$ the value of $F(t)$ follows from $F(t) + R(t) + B(t) = t$. Using I_A as the indicator for the event A, we have

$$\begin{aligned} B(t-1) &= B(t) - I_{\{v_t \in B(t)\}} - I_{\{v_t \in B(t),\, u_t \in B(t)\}} - I_{\{v_t \in R(t),\, u_t \in B(t)\}}, \\ R(t-1) &= R(t) - I_{\{v_t \in R(t)\}} + I_{\{v_t \in B(t),\, u_t \in B(t) \cup F(t)\}}. \end{aligned}$$

Taking expectations, conditional on $B(t)$, $R(t)$, and $F(t)$

$$\begin{aligned} \mathbf{E}B(t-1) &= B(t) - \frac{B(t)}{t} - \frac{B(t)}{t}\frac{B(t)-1}{t-1} - \frac{R(t)}{t}\frac{B(t)}{t-1} \\ \mathbf{E}R(t-1) &= R(t) - \frac{R(t)}{t} + \frac{B(t)}{t}\frac{B(t)-1}{t-1} + \frac{B(t)}{t}\frac{F(t)}{t-1}, \end{aligned}$$

and using $F(t) + R(t) + B(t) = t$, the unconditional expectations are

$$\mathbf{E}B(t-1) = \mathbf{E}B(t)\left(1 - \frac{1}{t} + \frac{1}{t(t-1)}\right) - \frac{\mathbf{E}[B(t)^2]}{t(t-1)} - \frac{\mathbf{E}[R(t)B(t)]}{t(t-1)}, \quad (1)$$

$$\mathbf{E}R(t-1) = \mathbf{E}R(t)\left(1 - \frac{1}{t}\right) + \frac{\mathbf{E}B(t)}{t} - \frac{\mathbf{E}[R(t)B(t)]}{t(t-1)}. \quad (2)$$

Size of the dominating set

Lemma 2 *Let $K > 0$ constant.*

(i) *For $X \in \{B, R\}$, and $t \in [n]$,*

$$\mathbf{Pr}\left(|X(t) - \mathbf{E}X(t)| \geq 2\sqrt{K(n-t)\log n}\right) \leq 2n^{-K}.$$

(ii) *Let $K > 0$ constant, then*

$$\mathbf{Pr}(|D_T - \mathbf{E}D(n)| \geq 2\sqrt{Kn\log n}) \leq 2n^{-K}.$$

Proof. Part (i) We use a standard Azuma martingale argument (see e.g. [10]).

Given a tree $T \in \boldsymbol{T}_n$, the sequence $s(T) = (u_n, u_{n-1}, ..., u_2)$ records the neighbour u_j of vertex v_j, when v_j was added to $T(j-1)$. All sequences s have the same measure. Let s and s' be two sequences differing only at the entry u_j for $j > t$. Thus

$$s(T) = (u_n, u_{n-1}, ..., u_{j+1}, x, u_{j-1}, ..., u_2), \qquad s'(T') = (u_n, u_{n-1}, ..., u_{j+1}, y, u_{j-1}, ..., u_2).$$

For different values of x and y this defines a partition of the set of all such sequences. Let $\mathcal{F}_j$ be σ-field generated by the sets $B_x = \{(u_n, u_{n-1}, ..., u_{j+1}, x, u_{j-1}, ..., u_2) : u_k \in \{1, \ldots, k-1\}, k < j\}$. Define $Z_j = \mathbf{E}\ (X(t) \mid \mathcal{F}_j)$. Clearly $Z_0 = \mathbf{E}X(t)$ and $Z_{n-t} = X(t)$. Azuma's inequality states that

$$\mathbf{Pr}(|X(t) - \mathbf{E}X(t)| > \lambda) \leq 2\exp\left(-\frac{\lambda^2}{2\sum_{j=1}^{n-t} c_j^2}\right),$$

where $c_j = |Z_j - Z_{j-1}|$. We next estimate c_j.

At any step $|B(t) - B'(t)| \leq 2$. In particular, for steps $t = n, ..., j+1$ the labeling is identical. At step j a difference may arise in the labeling of x, y. Let xPv_1, yQv_1 be the paths from x, y to the root v_1. The path P (e.g.) is the same in T, T'. For $t \leq j$ let $p(t), q(t)$ be x, y or the currently exposed leaves in these paths. Then $p(t), q(t)$ are the only vertices in $T(t), T'(t)$ whose labeling differs. The proof of (i) follows by putting $\lambda = 2\sqrt{K(n-t)\log n}$.

Part (ii). We use the same sequences s, s', as before. The beauty of the CGH Algorithm is that the size of the optimal dominating set is not affected by the order in which the leaves are removed. Thus we vary our normal application of the algorithm and first remove all leaves $v_n, ..., v_j$ followed by all leaves except x, y (when these vertices become leaves) in some fixed order. Up to this point the partially constructed dominating set D^* of T, T' is the same. Now we consider the remaining path $xQy = xw_1...w_ky$, where the labeling of x, y may differ in T, T' and the internal vertices Q (if any) have a fixed labeling resulting from our application of the algorithm up to this point.

Let L, L' be an optimal labeling of xQy in T, T' given its initial labeling. What happens when we change the labeling $\ell(x)$ of x in L to that $\ell'(x)$ in L'? Deleting x cannot increase the size of the dominating set in Qy; for if $x \in R$ relabel v_1 as R if necessary, to give a *feasible* labeling. Moreover it can decrease it by at most 1; for assuming it decreases it by at least 2, replacing x, labeled required gives a smaller feasible dominating set; a contradiction. Thus changing $\ell(x), \ell(y)$ to $\ell'(x), \ell'(y)$ alters the size of the dominating set between T, T' by at most 2. □

The size of the dominating set $\mathcal{D}$ of $T(n)$ obtained by the CGH Algorithm satisfies

$$D_{T(n)} = I_{\{v_1 \in B(1) \cup R(1)\}} + \sum_{t=2}^{n} I_{\{v_t \in R(t)\}},$$

and thus

$$\mathbf{E}D(n) = \mathbf{E}B(1) + \sum_{t=1}^{n} \frac{\mathbf{E}R(t)}{t}.$$

For any n, $d(n) = \mathbf{E}D(n)/n$ is a well defined variable, which we estimate.

Estimating $d(n)$. It would be interesting to obtain analytic solutions to $\mathbf{E}R$, and $\mathbf{E}B$, and hence $\mathbf{E}D$, from the recurrences (1), (2) above. However our attempts to do this lead to implicit functional relationships for these variables, which do not seem to be expressible explicitly as functions of t. Fortunately the recurrences (1), (2) are simple, and we resort to a numerical approach, based on accurate upper and lower bounds $X_U(t), X_L(t)$ for the variables $X(t) = \mathbf{E}B(t), \mathbf{E}R(t)$ at any step t.

Lemma 3 *For $n \geq 5 \times 10^8$, $d(n) = 0.3745...$*

Proof. It follows from Lemma 2(i) that with probability $1 - O(n^{-K})$, $B(t)$, and $R(t)$ are within $2\sqrt{Kn \log n}$ of the expected value. Choosing $K > 1$ constant, the bounds hold simultaneously for $t = 1, ..., n$ with probability $1 - O(n^{-K+1})$. Conditional on this, we can write

$$\mathbf{E}(B(t))^2 = (\mathbf{E}B(t) + \alpha_t)^2, \qquad \mathbf{E}(R(t)B(t)) = (\mathbf{E}R(t) + \beta_t)(\mathbf{E}B(t) + \alpha_t),$$

where $|\alpha_t|, |\beta_t| \leq \epsilon_t = 2\sqrt{K(n-t) \log n}$.

This leads to (e.g.)

$$B_L(t-1) = B_L(t)\left(1-\tfrac{1}{t}+\tfrac{1}{t(t-1)}\right) - \frac{(B_U(t))^2}{t(t-1)} - \frac{R_U(t)B_U(t)}{t(t-1)} - \frac{(3B_U(T)+R_U(t))\epsilon_t + 2(\epsilon_t)^2}{t(t-1)},$$
$$B_U(t-1) = B_U(t)\left(1-\tfrac{1}{t}+\tfrac{1}{t(t-1)}\right) - \frac{(B_L(t))^2}{t(t-1)} - \frac{R_L(t)B_L(t)}{t(t-1)} + \frac{(3B_L(T)+R_L(t))\epsilon_t + 2(\epsilon_t)^2}{t(t-1)},$$

which are well defined iterates which can be computed directly given the initial conditions $B_L(n) = B_U(n) = n, R_L(n) = R_U(n) = 0$, for $t \in [\delta n, n]$ and any $\delta > 0$.

Fixing δ let $D_Y = \sum_{t=\delta n}^{n} R_Y(t)/t$ for $Y = L, U$. Then

$$D_L(1-2n^{-K}) \leq \mathbf{E}D(n) \leq D_U + \delta n + n2n^{-K}.$$

The last entry follows, as n upper bounds the size of the dominating set of any tree in $\boldsymbol{T}_n$. Choosing $\delta = 0.00001$, $K = 2$ and $n = 5 \times 10^8$ the resulting value $d(n) = 0.3745...$ follows. $\square$

References

[1] K. Alzoubi, P. J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proc. 3rd ACM Internat. Symp. on Mobile Ad-hoc Networking & Computing*, pp 157–164, 2002.

[2] A. Barabási, and R. Albert. *Emergence of scaling in random networks*, Science, **286** (1999), 509–512.

[3] B. Bollobás. *Random Graphs*. Academic Press, 1985.

[4] E. Cockayne, S. Goodman, and S. Hedetniemi. *A linear algorithm for the domination number of a tree*, Information Processing Letters, **4** (1975), 41–44.

[5] W. Duckworth and M. Zito. Sparse hypercube 3-spanners. *DAM*, 103:289–295, 2000.

[6] U. Feige. A threshold of $\ln n$ for approximating set cover. *JACM*, 45:634–652, 1998.

[7] M. R. Garey and D. S. Johnson. Strong NP-Completeness results: Motivation, examples, and implications. *Journal of the Association for Computing Machinery*, 25(3):499–508, 1978.

[8] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, editors. *Domination in Graphs: Advanced Topics*. Marcel Dekker, 1998.

[9] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in Graphs*. Marcel Dekker, 1998.

[10] C. McDiarmid. On the method of bounded differences, *Surveys in Combinatorics 1989, LMS Lecture Note Series 141* (editor J. Siemons) 148-188, CUP (1989).

[11] R. Smythe, and H. Mahmoud. *A survey of recursive trees*, Theory of Probability and Mathematical Statistics, **51** (1996), 1–29.

[12] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Trans. Parallel and Dist. Systems*, 13:14–25, 2002.

[13] B. Wieland and A. P. Godbole. On the domination number of a random graph. *Elec. J. Combinat.*, 8:# R37, 2001.

[14] M. Zito. Greedy algorithms for minimisation problems in random regular graphs. *Proc 9th ESA*, pp 524–536, LNCS 2161. Springer-Verlag, 2001.