Bounds on the bisection width for random d-regular graphs

J. Díaz [a,1,2] M.J. Serna [a,*,1,3] N.C. Wormald [b,4]

[a] *Dept. Llenguatges i Sistemes, Universitat Politecnica de Catalunya Jordi Girona Salgado 1–3, 08034 Barcelona, Spain.*

[b] *Dept. Combinatorics and Optimization, University of Waterloo Canada N2L 3G1.*

Abstract

In this paper we provide an explicit way to compute asymptotically almost sure upper bounds on the bisection width of random d-regular graphs, for any value of d. The upper bounds are obtained from the analysis of the performance of a randomized greedy algorithm to find bisections of d-regular graphs. We provide bounds for $5 \leq d \leq 12$. We also give empirical values of the size of the bisection found by the algorithm for some small values of d and compare it with numerical approximations of our theoretical bounds. Our analysis also gives asymptotic lower bounds for the size of the maximum bisection.

Key words: Bisection width, Random regular graphs

1 Introduction

Given a graph $G = (V, E)$ with $|V| = n$ and n even, a *bisection* of V is a partition of V into two parts each of cardinality $n/2$, and its *size* is the number

* Corresponding author.
Email addresses: diaz@lsi.upc.es (J. Díaz), mjserna@lsi.upc.es (M.J. Serna), nwormald@uwaterloo.ca (N.C. Wormald).

1 Partially supported by the IST programme of the EU under contract IST-2001-33116 (FLAGS) and by the Spanish CICYT project TIC 2001-4197-E.
2 Partially supported by the *Distincio per a la Recerca 2002* of the Generalitat de Catalunya.
3 Partially supported by the Spanish CICYT project TIC 2002-04498-C05-03.
4 Supported by the Canada Research Chairs Program and partially by the Australian Research Council when this author was at the University of Melbourne.

of edges crossing between the parts. A *minimum bisection* (min bisection) is a bisection of V with minimal size. The decision problem associated to finding a minimum bisection is known to be NP-complete [13], even for 3-regular graphs [6]. The best approximation for min bisection on general graphs is a $O(\log^2 n)$ approximation ratio algorithm [10]. No better results are known in particular for d-regular graphs. On the other hand, several exact polynomial-time algorithms are known for particular graph instances (for a survey of results see [8]) and several nice heuristics are known for the problem (see for example [17,3]).

The size of a min bisection is called the *bisection width* and the *min bisection problem* consists of finding a minimum bisection in a given G. In the present paper, we give a family of randomized algorithms which give asymptotic upper bounds as $n \to \infty$ on the bisection width of almost all d-regular graphs, where d is fixed.

Plenty of results are known on bisection width. With respect to lower bounds, in 1975 Fiedler gave a spectral lower bound of $\lambda_2 n/4$ applicable for any graph, where λ_2 is the second eigenvalue of the Laplacian of the graph [12]. In 1984, Bollobás provided a lower bound of $(\frac{d}{4}-\frac{\sqrt{d\ln 2}}{2})n$, for almost all d-regular graphs [4]. Later Kostochka and Melnikov proved that almost all cubic graphs have bisection width greater than $0.101n$ [18]. Using spectral techniques, Bezrukov et al. gave lower bounds of $0.082n$ for the bisection width of cubic Ramanujan graphs, and of $0.176n$ for the case of 4-regular Ramanujan graphs [2].

Regarding upper bounds, Kostochka and Melnikov proved that asymptotically as $n \to \infty$, all d-regular graphs have bisection width of at most $\frac{d-2}{4}n + O(d\sqrt{n}\log n)$ [18]. Later, Alon proved that for $n > 40d^9$, all d-regular graphs have bisection width at most $(\frac{d}{4} - \frac{3\sqrt{d}}{32\sqrt{2}})n$ [1]. More recently, Monien and Preis [20] gave upper bounds on the bisection width of $(\frac{1}{6}+\epsilon)n$ for 3-regular graphs and of $(0.4+\epsilon)n$ for 4-regular graphs, for any $\epsilon > 0$, when n is larger than some function of the chosen ϵ. To the best of our knowledge, the most recent result on bisection width was given in [7], where it was proved that the bisection width of a random 4-regular graph on n vertices is asymptotically smaller than $(\frac{1}{3}+\epsilon)n$, with probability tending to 1 (a.a.s.). This result was proved by analysing a simple greedy algorithm, a variant of which yielded a bisection of a cubic random graph on n vertices of width asymptotically almost surely smaller than $0.174039n$.

The problem of finding the maximum bisection size has also received considerable attention. The max bisection problem is again NP-hard even for planar graphs [16]. Exact polynomial-time algorithms are known for particular graphs (for instance bounded treewidth graphs [16]). There are known constant approximation algorithms for the max bisection problem on general graphs [15]. Moreover, there is an approximation algorithm with a 0.795 ratio for d-regular graphs [11] and an approximation algorithm with a 0.9326 ratio for the particular case of cubic graphs [14].

The max bisection size is a lower bound on the maximum size (number of edges) of a bipartite subgraph. Locke [19] showed that a d-regular graph which is not complete or a cycle has a bipartite subgraph with at least $(nd/4)d/(d-1)$ edges if d is even, and at least $(nd/4)((d+1)/d + 2/d^2)$ edges if d is odd. Shearer [21] improved this result to $(nd/4)+n\sqrt{d}/8\sqrt{2}$ for triangle-free graphs, a property which a positive fraction of random regular graphs have. Our lower bounds for maximum bisection in random d-regular graphs easily exceed these bounds.

In Section 2 we present a basic randomized algorithm to find a (small) bisection of a graph by 2-colouring its vertices in a greedy way. The next vertex to be coloured is chosen according to a prioritisation scheme. The priority depends on to the status of a vertex with respect to the number of neighbours it has of either colour. Many different priority schemes were considered, each specified by a list of the types of vertices (i.e. their possible status with respect to colours of neighbours).

This prioritisation scheme is significant both as a simplification and as a generalisation of the method in [7], where only 3-regular and 4-regular graphs were considerd. It is a generalisation because, to each of the algorithms given there, there are corresponding algorithms of the general type considered in this paper which have equivalent asymptotic performance (although the algorithms do not give identical results). It is a simplification, because the method in [7] was to specify one or two main phases of the algorithm. In each main phase, two types of vertex were coloured, with one of the specified types having priority. Such detailed control of the algorithm is difficult to generalise to higher d because of the difficulty of knowing which types of vertices might be available when one phase ends and a new phase starts. The key idea in the present paper is to specifiy which types of vertices should have priority over which others, throughout the whole algorithm. The transitions between phases then become automatic. It is hard to substantiate this claim without looking at the algorithm in more detail, since even the definition of a phase becomes more delicate with this approach. A similar effect occurred in the analysis of greedy algorithms for finding independent sets in random regular graphs [23], but the situation there was considerably simpler. In that case, the prioritisation was merely according to the degree of a vertex during a deletion algorithm, whilst in the present case, the best prioritisation list is much harder to determine. Moreover, in the present case, the algorithm in some sense returns to phase which it visited earlier, and this did not happen in [23].

In Section 3, we begin the analysis of the performance using the differential equation method. Full justification is not included here, though we have no doubt that the differential equations give a correct picture of these algorithms. For any given d, we choose an appropriate priority list, set the equations and solve them numerically to find the asymptotic bisection width for the random d-regular graphs under consideration. In the same section, we produce empirical evidence indicating that there are two types of optimal priority lists of the vertices: one for even values of d and the other for odd values of d. In Section 4, we define an associated deprioritised algorithm which we then analyse and show that the results obtained in Section 3 correctly apply for such algorithms. In Section 5, we give empirical results comparing the values obtained numerically from the differential equations with the bisection width obtained by the randomized greedy algorithm. In Section 6, we discuss the maximum bisection results. It should be emphasised that the main contribution of this paper is to give better asymptotic bounds for the bisection width of d-regular graphs ($d > 4$), and the algorithm produced in Section 2 is only of methodological value.

2 The priority-greedy algorithm for random d-regular graphs

In this section, we describe a family of randomized greedy procedures to find a bisection for d-regular graphs. We also introduce some generic notation to be used later.

Given a graph, and given a partial assignment of colours red (R) and blue (B) to its vertices, we classify the uncoloured vertices according to the colours of their neighbours:

> An uncoloured vertex is of *Type* (r, b), if it has r neighbours coloured R and b neighbours coloured B.

For $r \leq b$, we say that a pair of uncoloured vertices is a *symmetric pair* if their types are (r, b) and (b, r) for some r and b. We then call this an (r, b)-*symmetric* pair, or a symmetric pair of *type* (r, b).

The greedy procedure works by colouring vertices chosen randomly in symmetric pairs, to maintain balancedness, and repeatedly uses the *majority operation* (Maj), that colours each vertex of an (r, b)-symmetric pair, $r < b$, with the majority colour among its neighbours, and, given an (r, b)-symmetric pair with $r = b$, randomly colours one vertex of the pair R and the other B .

We assume that the symmetric pair types have the priorities $0, 1, 2, \ldots$ associated with them (a larger number denotes higher priority). The priority-greedy algorithm for random d-regular graphs is given in Figure 1.

Initial step:	input $\texttt{prio}(r, b)$ for all $r \leq b \leq d$; select two non-adjacent vertices u.a.r., colour one with R and the other with B.
Main iteration:	

	while there is at least one uncoloured symmetric pair
	    do
	        let (r, b) denote the highest priority type of
	          an uncoloured symmetric pair;
	        select u.a.r. an (r, b)-symmetric pair
	          and perform Maj;
	end while

Clean up:	colour any remaining uncoloured vertices, half of them R and half B, in any manner, and output the bisection R, B.

Fig. 1. Algorithm priority-greedy for obtaining a bisection of a d-regular graph

This algorithm takes as input a predetermined priority list assigning a distinct priority, $\texttt{prio}(r, b)$, to each symmetric pair type (r, b). We impose the conditions on all priority lists that $\texttt{prio}(0, 0) = 0 < \texttt{prio}(r, b)$ whenever $(r, b) \neq (0, 0)$. Note that the priority of pairs (r, b) with $r + b = d$ is immaterial since colouring vertices in such pairs cannot affect the remainder of the algorithm. So for simplicity we assume that all such vertices have negative priority, and only those with $r + b < d$ need to be specified.

3 Analysis: The differential equation system derived from the priority-greedy algorithm

We follow the description in [7], extending it to the d-regular setting for arbitrary d. The algorithms considered there give equivalent results in special cases of the priority-greedy algorithm, for particular priority lists and for $d = 3$ and 4. (Notice that the algorithms described in [7] for $d = 3$ and $d = 4$ are different from the *general* algorithm presented in this paper)

One method of analysing the performance of a randomised algorithm is to use a system of differential equations to express the expected changes in the variables describing the state of the algorithm during its execution. An exposition of this method can be found in [24], which includes various examples of graph-theoretic optimisation problems.

We use the pairing model to generate n-vertex d-regular graphs u.a.r. Briefly, to generate such a random graph, it is enough to begin with dn points in n cells, and choose a random perfect matching of the points, which we call a *pairing*. The corresponding pseudograph (possibly with loops or multiple edges) has the cells as vertices and the pairs as edges. Any property a.a.s. true of the random pseudograph is also a.a.s. true of the restriction to random graphs, with no loops or multiple edges, and this restricted probability space is uniform (see for example [5,25] for a full description).

We consider the priority-greedy algorithm applied directly to the random pairing. As discussed in [24], the random pairing can be generated pair by pair, and at each step a point p can be chosen by any rule whatsoever, as long as the other point in the pair is chosen u.a.r. from the remaining unused points. We call this step *exposing* the pair containing p. At each step of the priority-greedy algorithm in which a vertex is coloured, we expose all pairs containing points in that vertex.

Algorithms like this one typically pass through a number of phases which cause some complications for analysis. Part of the complication is caused by the prioritisation. There is a way of deprioritising the algorithm expounded in [26] which makes analysis easier and does not alter the asymptotic result. The following discussion serves both as an informal justification of the results for the priority-greedy algorithm and also as a rigorous preparation for the analysis of the corresponding deprioritised algorithm which we introduce later in this section.

Informally speaking, in a typical part of the algorithm, there will be symmetric pairs of one particular type, (r_0, b_0), which are plentiful in the graph and are quite regularly chosen in the main iteration of the algorithm. Symmetric pairs of types with higher priority may also be regularly chosen, but will be rare and the number of such pairs in the graph will regularly drop to 0 (otherwise those of type (r_0, b_0) would *not* be used regularly). In this situation, we say that (r_0, b_0) is the *basic* type. The algorithm will typically pass through *phases*, determined by points at which, roughly speaking, the basic type changes. A phase finishes when either symmetric pairs with higher priorities than the current basic type become plentiful, or those with the current basic type become very scarce. The boundaries of the phases are best defined precisely in terms of the solution of a set of differential equations which we now proceed to derive.

At each point in the algorithm, let $Z_{r,b}$ denote the number of uncoloured vertices of type (r, b) (so that $Z_{r,b} = Z_{r,b}(t)$ where t is the time, or the index of the step in the algorithm), and let W denote the number of points not yet involved in exposed pairs. Then

$$W = \sum_{r+b<d} (d - r - b) Z_{r,b}. \tag{1}$$

From this point onwards, we assume the reader is familiar with the argument in [7], and omit some of the elaborate justifications which are identical to those appearing there. Let $d_{r,b}$ denote the expected contribution to $\Delta Z_{r,b}$, the increment of $Z_{r,b}$, due to exposing the pair containing a point in a vertex u which has just been coloured red by the priority-greedy algorithm. Then the probability that the other point chosen in the pair is in a vertex v of type (i, j) is $(d-i-j)Z_{i,j}/(W-1)$ (except for a correction of size $O(1/W)$ due to the change in status of u). Hence, ignoring terms of size $O(1/W)$,

$$d_{r,b} = \frac{\alpha_{d+1,r+b}Z_{r-1,b} - \alpha_{d,r+b}Z_{r,b}}{W}$$

for $r + b \le d$, where

$$\alpha_{x,y} = \begin{cases} x-y & \text{if } x > y, \\ 0 & \text{otherwise.} \end{cases}$$

In the following we continue to ignore terms of size $O(1/W)$. The equations due to the case that u is coloured blue are

$$d_{r,b} = \frac{\alpha_{d+1,r+b}Z_{r,b-1} - \alpha_{d,r+b}Z_{r,b}}{W}$$

for $r + b \le d$. Letting $\bar{d}_{r,b}$ be the sum of these two expected increments, from the two vertices in the symmetric pair, we have

$$\bar{d}_{r,b} = \frac{\alpha_{d+1,r+b}(Z_{r,b-1} + Z_{r-1,b}) - 2\alpha_{d,r+b}Z_{r,b}}{W}, \tag{2}$$

for $r + b \le d$.

We now make a definition which is not rigorously stated for the priority-greedy algorithm but becomes rigorous in the context of deprioritised algorithms (as explained later). In fact, from here onwards, the discussion may be taken entirely as motivation for the definition of the deprioritised algorithms, which will be analysed separately. For $r \le b$, let $\phi_{r,b}$ denote the probability of processing an (r, b)-symmetric pair at some step in a given phase. (This is not a measurable probability for the priority-greedy algorithm since it depends on the state of the algorithm at the time; however it will become part of the *definition* of the deprioritised algorithm.) Assume that at the beginning of a new phase, (r_0, b_0) is the basic type, and that symmetric pairs of this type are plentiful. Thus, for a considerable part of the algorithm, no vertices of lower priority will be chosen for Maj. We calculate the ϕ's for the basic symmetric pair type and all other types with higher priority.

Let $\mathcal{B}^-$ denote the set of types of symmetric pairs with strictly greater priority than the basic one, (r_0, b_0), and let $\mathcal{B} = \mathcal{B}^- \cup \{(r_0, b_0)\}$.

Given the assumption about the ϕ's, the expected number of points in a blue (or red) vertex when Maj is perfomed is

$$c = \sum_{(r',b')\in\mathcal{B}} (d - r' - b')\phi_{r',b'}. \tag{3}$$

Since the types in $\mathcal{B}^-$ are used up as fast as they appear, we *expect*

$$\phi_{r,b} = c\beta_{r,b}\bar{d}_{r,b}, \qquad (r,b) \in \mathcal{B}^- \tag{4}$$

where $\beta_{r,b}$ is 1 if $r \neq b$, and $\frac{1}{2}$ if $r = b$. The function $\beta_{r,b}$ appears here because, in the case that an (r,r)-symmetric pair is processed, there are *two* vertices of that type being coloured, so the number of times the operation is required to balance the other changes in the numbers of such vertices is halved. In addition, we are assuming here that the vertices of both types in a given symmetric pair are used up at approximately equal rates.

Of course there is an extra equation involving (r_0, b_0):

$$\sum_{(r,b)\in\mathcal{B}} \phi_{r,b} = 1. \tag{5}$$

Using this equation to eliminate ϕ_{r_0,b_0} from (3), and then substituting out all other $\phi_{r,b}$ using (4), we can easily solve for c to obtain

$$c = \frac{d - r_0 - b_0}{T}, \qquad T = 1 + \sum_{(r',b')\in\mathcal{B}^-} (r' + b' - r_0 - b_0)\beta_{r'b'}\bar{d}_{r',b'}. \tag{6}$$

Now $\phi_{r,b}$ is determined from (4) for $(r,b) \in B^-$, and a little computation produces from (5)

$$\phi_{r_0,b_0} = \frac{1}{T} + \frac{1}{T}\sum_{(r',b')\in B^-} (r' + b' - d)\beta_{r'b'}\bar{d}_{r',b'}. \tag{7}$$

Assuming validity of the equations for the ϕ's, the expected increments of the random variables $Z_{r,b}$ at each iteration are given for $r \leq b$ by

$$\mathbf{E}\left[\Delta(Z_{r,b})\right] = c\bar{d}_{r,b} - (1 + \delta_{r,b})\phi_{r,b} \tag{8}$$

where $\delta_{r,b}$ is the Kronecker delta (1 if $r = b$, 0 otherwise). The terms subtracted are due to the change in types of the symmetric pair of vertex being coloured; in the case $r = b$, two vertices of type (r,r) are lost. Note that since $(1 + \delta_{r,b})\beta_{r,b} = 1$, this expected change is 0 whenever $(r,b) \in \mathcal{B}^-$, as required. For $r > b$ we have similarly

$$\mathbf{E}\left[\Delta(Z_{r,b})\right] = c\bar{d}_{r,b} - (1 + \delta_{r,b})\phi_{b,r}. \tag{9}$$

As done in [7] for the case $d = 4$, we may express the above expected increments as a set of differential equations for some approximating variables. With $\mathbf{z}$ denoting the vector of $z_{r,b}$, $0 \le r \le b \le d$, we scale both time and the variables by dividing by n, and approximate $Z_{r,b}/n$ by $z_{r,b}$ and t/n by x. Since the scaling is by the same factor in both cases, each $\mathbf{E}[\Delta(Z_{r,b})]$ approximates the differential $z'_{r,b}$. Representing ϕ by θ, T by $\gamma(\mathbf{z})$, $\bar{d}_{r,b}$ by $g_{r,b}(\mathbf{z})$ and W/n by $w(\mathbf{z})$, the equations suggested by (8) are, in view of (6),

$$z'_{r,b} = \frac{d - r_0 - b_0}{\gamma(\mathbf{z})} g_{r,b}(\mathbf{z}) - (1 + \delta_{r,b})\theta_{r,b} \tag{10}$$

for $r \le b$ and $r + b \le d$, where from (4) and (6), (7), (6), (2) and the definition of W, respectively,

$$\begin{aligned}
\theta_{r,b}(\mathbf{z}) &= \frac{(d - r_0 - b_0)\beta_{r,b} g_{r,b}(\mathbf{z})}{\gamma(\mathbf{z})}, \qquad (r,b) \in \mathcal{B}^-, \\
\theta_{r_0,b_0}(\mathbf{z}) &= \frac{1}{\gamma(\mathbf{z})} + \frac{1}{\gamma(\mathbf{z})} \sum_{(r',b') \in \mathcal{B}^-} (r' + b' - d)\beta_{r'b'} g_{r',b'}(\mathbf{z}), \\
\theta_{r,b}(\mathbf{z}) &= 0, \qquad (r,b) \notin \mathcal{B}, \\
\gamma(\mathbf{z}) &= 1 + \sum_{(r',b') \in \mathcal{B}^-} (r' + b' - r_0 - b_0)\beta_{r'b'} g_{r',b'}(\mathbf{z}), \\
g_{r,b}(\mathbf{z}) &= \frac{\alpha_{d+1,r+b}(z_{r,b-1} + z_{r-1,b}) - 2\alpha_{d,r+b} z_{r,b}}{w(\mathbf{z})}, \\
w(\mathbf{z}) &= \sum_{r+b \le d} \alpha_{d,r+b} z_{r,b}.
\end{aligned}$$

Here, in the definition of $g_{r,b}$ and w, for $r > b$ we define $z_{r.b} = z_{b,r}$. Since the equations are symmetric when the indices are swapped over, there is no need to keep track of the variables $z_{r,b}$ for $r > b$.

The increase in the size of the bisection due to a vertex of type (r,b) being coloured blue is r (this always occurs in Maj if $r < b$, whilst if $r = b$ it makes no difference). The symmetric vertex being coloured red, and of type (b,r), also increases the bisection by r. Thus, the expected increase per algorithm step is $2\sum_{(r,b)\in\mathcal{B}} r\,\phi_{r,b}$. Letting z denote the bisection size (divided by n), this suggests the equation

$$z' = 2 \sum_{(r,b) \in \mathcal{B}} r\,\theta_{r,b}. \tag{11}$$

Equations (10,11) are the differential equations for a phase with (r_0, b_0) being the basic type. The definition of phases is inductive in terms of the differential equations. For phase 1, the basic type is $(r_0, b_0) = (1, 0)$. Phase k will end when either $\theta_{r_0,b_0} = 0$, in which case the basic type for phase $k+1$ will have priority $\texttt{prio}(r_0, b_0) + 1$, or when z_{r_0,b_0} begins to go negative, in which case, the priority of the basic type (r, b) in the next phase will be initially set equal to whichever type has highest priority among those with $z_{r,b} > 0$. (If all z are 0, the process ends.) If the end-of-phase criteria immediately apply to the new basic type, then the next basic type is immediately redefined using the same rule, so each phase has non-zero length unless these rules cause the definition of basic type to be passed around in a cyclic fashion amongst basic types each of which satisfies the criteria of the end of a phase. This seems unlikely to happen much before the end of the process, but is not necessary to rule out at this stage as it can be ruled out with numerical computations. There is also the possibility that both criteria are met simultaneously; in this case we somewhat arbitrarily choose to follow the θ test first; that is, increase the priority of the basic variable. It seems reasonable that the priority-greedy algorithm will spend time in a phase processing higher priority vertices if θ_{r_0,b_0} becomes 0, but we do not offer a proof of this since we do not attempt here to show that these differential equations describe the priority-greedy algorithm. Instead, we merely use these rules and differential equations to define the phases, and these will be used below in defining the deprioritised algorithm.

The variables' initial conditions at the start of a phase are just their values at the end of the previous phase. The whole calculation begins with all variables equal to 0, except for $z_{0,0} = 1$. The size of the bisection is represented by the value of z (scaled up by a factor n) when $w(\mathbf{z})$ reaches 0, which must happen when the values of all the $z_{r,b}$ reach 0 simultaneously. The last few phases are (at least in the cases calculated below) those in which the basic type (r_0, b_0) has $r_0 + b_0 = d$. Since the changes in the variables during these phases are determinisitic, they can be skipped in practice if the appropriate quantity is added to z. We believe that the value of z accurately represents the size of the bisection obtained by the priority-greedy algorithm, although we prove such a statement only for the deprioritised algorithm below. Before that, it is interesting to look at the results.

After trying many different priority lists, solving the resulting system of differential equations using a Runge-Kutta method of order 2, we focussed on two priority lists which appear to give the best results. The following determine the order amongst those (r, b) with $r + b < d$:

List A: $\texttt{prio}(i, j) > \texttt{prio}(k, l)$ iff $j - i < l - k$ or $(j - i = l - k$ and $i > k)$.
List B: Same as List A but swapping $\texttt{prio}(0, 2)$ with $\texttt{prio}(\lfloor d/2 \rfloor - 1, \lfloor d/2 \rfloor)$.

For example, with $d = 5$, List A places the types in the following order: (0,0), (1,1), (2,2), (0,1), (1,2), (0,2), (1,3), (0,3), (0,4); List B puts (0,2) before (1,2) but retains all other relative rankings.

List A appears, from the results of the calculations, to perform better for d odd and List B performs better for d even. However, for larger d, this is not clearly

d	5	6	7	8	9	10	11	12
list A	0.5028	0.6675	0.8502	1.0391	1.2317	1.4278	1.624	1.823
list B	0.5247	0.6674	0.8590	1.0386	1.2318	1.4278	1.624	1.823
min(A,B)	0.5028	0.6674	0.8502	1.0386	1.2317	1.4278	1.624	1.823

Table 1
Results for lists A and B, rounded up

demonstrated to the accuracy with which we can confidently quote the results (due to errors inherent in numerical solution of the differential equations). The bounds obtained with Lists A and B for $d \leq 12$ are given in Table 1. The experiments were performed on a PC with a Pentium III processor. Machine power was too limiting to go much further than this with sufficient accuracy, but the further digits obtained, which we do not report here, suggested that the ranking of Lists A and B according to the parity of d continues, at least up to $d = 12$.

4 The deprioritised algorithm

In this section we use the approach in [26] to define and analyse the deprioritised algorithms which enable us to deduce the bounds on bisection width described in Table 1.

A major complicating factor for analysis of prioritised algorithms, such as priority-greedy, is that the rate of change of variables is not smooth: when a vertex of one type is coloured, the effects are different from that of another type being processed. To avoid this, we define a general deprioritised algorithm as follows. Recall that $Z_{r,b}(t)$ is the number of uncoloured vertices of type (r, b) at time t. Let $\mathbf{Z}(t)$ denote a vector containing all such variables. In the following, $\phi(r, b, t, \mathbf{Z})$ is any predefined function such that for each fixed t and $\mathbf{Z}$, the values of $\phi(r, b, t, \mathbf{Z})$ define a probability function over the (r, b) which have $r \leq b$.

We also extend the definition to allow a period of time to be specified in which the deprioritised algorithm does nothing. This is merely to simplify analysis, and we refer to it as a *static* period.

The definition of the appropriate function ϕ, and the proof of the desired approximation, is inductive. The inductive step is provided by the following claim. Let $\mathbf{z}$ denote the solution to the differential equations described in the previous section, and let x_k denote the x value at the start of phase k (or, if $k-1$ is the final phase, x_k is its end). Also let $t_k = \lfloor nx_k \rfloor$.

Initial step:	input ϕ;
	set $t = 0$;
Main iteration:	repeat the following until "stop" is reached:
	choose (r, b) according to probability $\phi(r, b, t, \mathbf{Z})$
	if no symmetric pair of type (r, b) exists, stop;
	else select u.a.r. a symmetric pair of type (r, b)
	and perform Maj;
Clean up:	colour any remaining uncoloured vertices, half of them R
	and half B, in any manner, and output the bisection R, B.

Fig. 2. General deprioritised greedy algorithm for bisection width of a d-regular graph

We say that $\mathbf{Z}(t)$ is *δ-approximated by* $\mathbf{z}(x)$ if for all $r \le b$ we have $|Z_{r,b}(t) - nz_{r,b}(x)| < \delta n$ and $|Z_{b,r}(t) - nz_{r,b}(x)| < \delta n$. The following result refers to the particular function $\mathbf{z}$ which solves (10) with initial conditions as discussed above.

Lemma 1 *For all $\epsilon > 0$ there exists $\delta > 0$ and a function ϕ such that for the corresponding deprioritised algorithm, a.a.s. if $\mathbf{Z}(t_k)$ is δ-approximated by $\mathbf{z}(t_k/n)$ then $\mathbf{Z}(t_{k+1})$ is ϵ-approximated by $\mathbf{z}(t_{k+1}/n)$.*

Proof We first make sure there is a plentiful supply of vertices of each type $(r, b) \in \mathcal{B}^-$ for which $\theta_{r,b} > 0$ at some point during phase k. Let S denote the set of such types. For $(r, b) \in S$, it must be that $g_{r,b} > 0$ at some point in the phase. This necessitates either $z_{r,b-1} > 0$ or $z_{r-1,b} > 0$ at that point. If $z_{r',b'} = 0$ for all types (r', b') with $r' \le r$ and $b' \le b$ at the start of the phase (i.e. at time t_k), then $g_{r',b'} \le 0$ for all these types and so the derivatives of these variables must remain zero throughout this phase, a contradiction. So at the start of the phase (i.e. $x = x_k$) $z_{r',b'} > 0$ for some type (r', b') with $r' \le r$ and $b' \le b$. Choose one such type (r', b') for each $(r, b) \in S$, and let S' denote the set of types (r', b') so chosen. It is permissible for one type (r', b') to serve two types (r, b).

Now we turn to defining a function ϕ for the initial stage of the deprioritised algorithm, and we do this for some $\delta > 0$ to be chosen later, along the way noting some upper bounds on δ. Let $\hat{z}$ denote the minimum value of $z_{r',b'}(x_k)$ over all $(r', b') \in S'$. At the beginning of the phase, the deprioritised algorithm performs a *prephase* of $\lfloor n\hat{z}\delta'/(4d + 4)\rfloor$ steps with $\phi(r', b', t, \mathbf{Z}) = 1/|S'|$ for each $(r', b') \in S'$ and $\phi(r', b', t, \mathbf{Z}) = 0$ otherwise. We specify $\delta < \hat{z}/4$, and specify the constant δ' later. By the hypothesis of the claim, there are sufficient symmetric pairs of each type for the steps in the prephase to be carried out without running into the stop condition in the main iteration, since at most $2d + 2$ vertices change type in any one step.

We may apply [Theorem 3][26] (see [24] for a full description of the method), to the random variables $Z_{r,b}$ whose expected changes in one step are given, as in the argument above, by (8) and (9). We deduce from this result that the variables $Z_{r,b}$ will a.a.s. follow, with error $o(n)$, the trajectory of the differential equations determined as given by (10) but with $\theta_{r,b}$ set equal to 0 apart from $\theta_{r',b'} = 1/|S'|$ for each $(r',b') \in S'$. Again there is a symmetry in swapping the indices. The initial conditions of the equations are given by $Z_{r,b}(t_k)/n$.

If we instead change the initial values to $z_{r,b}(x_k)$ for all r and b, the error introduced is by hypothesis at most δ. Standard theory of first order differential equations implies that the error this produces at the end of the prephase is at most $h(\delta)$ for some function $h \to 0$ as $\delta \to 0$.

It follows that a.a.s. at the end of the prephase there are $nf_{r,b}+O(nh(\delta))+o(n)$ vertices of each type (r,b), for some function $f_{r,b}$ determined by the differential equations. By taking δ and δ' sufficiently small, it follows that for any desired $\epsilon_0 > 0$, a.a.s. $\mathbf{Z}(t_k + \delta')$ is ϵ_0-approximated by $\mathbf{z}((t_k + \delta')/n)$.

We need one more observation before leaving the prephase. It is easy to see from the definition of S' that the derivatives of all variables $z_{r,b}$ for $(r,b) \in S$ are strictly positive after the beginning of the prephase. Thus $f_{r,b} > 0$ for all such types, and so by taking δ much smaller than δ', we may ensure that a.a.s. $Z_{r,b} \geq nf_{r,b}/2$.

After the prephase, $\phi(r,b,t,\mathbf{z})$ is defined using equations (4), (6) and (7) (so t does not enter its formula, but the $z_{r,b}$ do, however these must be viewed for the purpose of defining ϕ as arbitrary indeterminates and not the solution to the earlier differential equation). Then of course $\phi(r,b,t,\mathbf{Z})$ is used as the probability in the main iteration in the algorithm. The expected changes in the values of the $Z_{i,j}$ in one step can then be calculated (asymptotically), given their values at the beginning of the step, and the result is (8) and (9) (again ignoring the $O(1/W)$ terms). All of the derivation of the differential equation (10) is now valid, although the initial conditions are determined by the (stochastic) values of the variables after the pre-phase.

Note that the definitions of ϕ, and through it the $\theta_{r,b}$, ensure that $z'_{r,b} = 0$ for all $(r,b) \in \mathcal{B}^-$. Arguing as for the prephase, the variables $Z_{r,b}$ will a.a.s. follow, with error $o(n)$, the trajectory of the differential equations determined by (10), but with these new initial conditions, until either the end of the phase is reached or one of the variables reaches 0. This a.a.s. does not happen with $Z_{r,b}$ for $(r,b) \in \mathcal{B}^-$ since it follows the solution, which remains a non-zero constant, with error $o(n)$. It may happen with (r_0,b_0), but not until within $o(n)$ of $n\hat{x}$, where $\hat{x}$ is the point at which the differential equation solution for z_{r_0,b_0} reaches 0. By taking ϵ_0 sufficiently small, $\hat{x}$ can be made as close as desired to x_{k+1}. We may then define a static period until t_{k+1}, and take ϵ_0 small enough to ensure that a.a.s. $\mathbf{Z}(t_{k+1})$ is ϵ-approximated by $\mathbf{z}(t_{k+1}/n)$.

The argument requires a minor variation for a phase in which $w(x)$ is not bounded away from 0. This must necessarily be the last phase, as all variables must tend to 0. The derivatives given in the right hand side of (10) do not then have bounded derivatives in the domain of interest. In this case we may terminate our examination of the algorithm at time $t_{k+1} - n\epsilon_0$. The changes to the variables $Z_{r,b}$ during a further $n\epsilon_0$ steps are $O(n\epsilon_0)$. So the same conclusion holds. Note that the denominator $\gamma(\mathbf{z})$ cannot approach 0 since it represents T in (6), and as seen from the formula for c, which has non-zero numerator, T is bounded below because c is bounded above. The latter follows from (3) since the sum of the ϕ's is 1. This completes the proof of the lemma. □

Using Lemma 1 and induction, we immediately obtain the following result.

Theorem 1 *Let k_0 be any fixed integer for which phase k_0 exists and let $\delta > 0$ be given. Then there is a $\phi(r, b, t, \mathbf{z})$ such that in the associated deprioritised algorithm, a.a.s. $\mathbf{Z}(t_k)$ is δ-approximated by $\mathbf{z}(t_k/n)$ for $k = 1, 2, \ldots, k_0$.*

The same argument also applies to the variable z with differential equation (11). Thus, $nz(x_{k_0}) + o(n)$ is a.a.s. an upper bound on the size of the bisection width of a random d-regular graph. The results on $z(x_{k_0})$ are described above. Note that one does not need to prove that a phase k_0, where numerical computations are terminated, is the final phase. All that is needed is to verify (largerly numerically) that the correct phases are computed, with all variables staying in the appropriate domain, up to phase k_0, which in all cases considered here was a phase with $r_0 + b_0 = d$. As mentioned above, the size of the final bisection can be computed by stopping at such a point.

An example of what could potentially cause a difficulty with this approach is that the value of z_{r_0,b_0} could remain very close to 0 at the start of a phase. Then the numerical evidence may not distinguish whether this phase continued or another should begin, thereby introducing large uncertainties into the results. Luckily, for all the results reported in this paper, each phase which lasted long enough to affect the results was such that the results verified its validity: if z_{r_0,b_0} and z'_{r_0,b_0} are both close to 0 at the start of a phase then z''_{r_0,b_0} (which can be computed from the differential equations) has a appreciable positive value, so that even allowing for the numerical errors, it must be positive. Similarly, at the start of each phase either ϕ_{r_0,b_0} is appreciably greater than 0, or the phase terminates immediately on the basis of the numerical calculation.

5 The experimental upper bounds

We have also generated a set of d-regular graphs for each $d = 5$ to 12, following the method described in [22]. We repeated the algorithm 10 times on each of the graphs, with priorities given by List A for d odd and List B for d even. The results, for five graphs with 10^5 vertices and two graphs with 2×10^5 vertices,

		e5-1	e5-2	e5-3	e5-4	e5-5	av:e5	2e5-1	2e5-2	av:2e5
d=5	avg	0.5046	0.5087	0.5042	0.5046	0.5038	0.5052	0.5036	0.5040	0.5038
	max	0.5060	0.5463	0.5060	0.5061	0.5060		0.5045	0.5052	
0.5028	min	0.5026	0.5026	0.5019	0.5035	0.5025		0.5032	0.5031	
d=6	avg	0.6692	0.6710	0.6695	0.6690	0.6689	0.6695	0.6687	0.6691	0.6689
	max	0.6719	0.6849	0.6720	0.6724	0.6711		0.6692	0.6703	
0.6674	min	0.6675	0.6682	0.6670	0.6672	0.6675		0.6677	0.6683	
d=7	avg	0.8517	0.8515	0.8517	0.8522	0.8516	0.8517	0.8511	0.8516	0.8513
	max	0.8536	0.8534	0.8541	0.8530	0.8545		0.8533	0.8524	
0.8502	min	0.8443	0.8482	0.8502	0.8511	0.8498		0.8496	0.8509	
d=8	avg	1.0410	1.0397	1.0403	1.0406	1.0404	1.0407	1.0407	1.0402	1.0404
	max	1.0438	1.0420	1.0426	1.0427	1.0422		1.0432	1.0416	
1.0386	min	1.0388	1.0378	1.0384	1.0379	1.0378		1.0392	1.0384	
d=9	avg	1.2340	1.2339	1.2336	1.2333	1.2335	1.2336	1.2325	1.2342	1.2333
	max	1.2374	1.2366	1.2353	1.2352	1.2357		1.2331	1.2364	
1.2317	min	1.2312	1.2306	1.2313	1.2308	1.2309		1.2316	1.2326	
d=10	avg	1.4292	1.4303	1.4289	1.4296	1.4300	1.4296	1.4291	1.4288	1.4289
	max	1.4326	1.4327	1.4299	1.4315	1.4333		1.4300	1.4301	
1.4278	min	1.4265	1.4293	1.4275	1.4280	1.4281		1.4283	1.4272	
d=11	avg	1.6257	1.6256	1.6256	1.6256	1.6262	1.6258	1.6251	1.6254	1.6254
	max	1.6276	1.6276	1.6286	1.6273	1.6288		1.6264	1.6270	
1.624	min	1.6227	1.6234	1.6236	1.6240	1.6239		1.6243	1.6240	
d=12	avg	1.8245	1.8248	1.8241	1.8241	1.8257	1.8246	1.8241	1.8244	1.8244
	max	1.8264	1.8273	1.8270	1.8260	1.8273		1.8248	1.8254	
1.823	min	1.8231	1.8214	1.8226	1.8226	1.8223		1.8237	1.8236	

Table 2
Size of the bisection obtained by the greedy algorithm for five graphs with $n = 10^5$ (e5-*) and two graphs with $n = 2 \times 10^5$ (2e5-*). The asymptotic almost sure upper bound from the differential equation analysis is given in the left column.

for each $d = 5, \ldots, 12$ are summarized in Table 2. The left column of the table includes the bound via the differential equations. The mean, max and min of the bisection values obtained using Algorithm 1 are given for each graph, and the means are also averaged over all graphs of each of the two sizes. [5]

[5] For any reader interested in checking the experiment, the graphs generated can be found at:
http://www.lsi.upc.es/~mjserna/dregraphs.html

6 Maximum Bisection

Let us consider the variation of the priority-greedy algorithm obtained replacing the majority operation (Maj) with the minority operation, that asigns to a vertex the colour minority among its coloured neighbors. Let us call this variation max priority-greedy.

Define an edge to be *fully coloured* when both its ends are finally coloured. A fully coloured edge is *monocoloured* if both ends have the same colour and *bicoloured* if both ends have different colour. So the edges monocoloured by priority-greedy get bicoloured by max priority-greedy and vice versa, whenever the vertices of the graph are treated in the same order (which happens with the same probability, in both cases) and the vertices that are not coloured by a Maj/Min operation get opposite colors (which again happens with the same probability). That is, every edge that counts in the bisection for one algorithm does not count in the other and vice versa. So, taking into account that the total number of edges in a d-regular graph is $dn/2$, we have the following complementary bounds for the maximum bisection: the size of the maximum bisection in a random d-regular graph is a.a.s. at least $dn/2 - c_d n$, where c_d is the min value given in Table 1 in column d. (Note that c_d *greater* than the best possible constant, produced from our argument, in both cases.)

7 Conclusions

As mentioned above, in [7] there is an analytic expression for the bound obtained on the bisection width of a random 4-regular graph, obtained from differential equations corresponding to the priority-greedy algorithm. When run for $d = 4$, Lists A and B give the same theoretical result, because the types $(0, 2)$ and $(1, 2)$ never become basic: there is only one phase, with $(0, 1)$ basic (see [7], where the algorithm is expressed in a different way but gives the same differential equations).

In this paper we have proposed a randomized greedy procedure which bounds the bisection width of any d-regular graph, and analyzed its typical performance on random d-regular graphs. The algorithm uses a predefined list of priorities. Furthermore, a related algorithm shows complementary bounds for the maximum bisection size. We give a proof that for any given list, and any $d \geq 3$, the values of the size of the bisection obtained by the related deprioritised algorithm are concentrated around the value determined by the solution of a set of differential equations.

Experimentally, we notice that a good list of priorities is given by List A for

$d \geq 6$ even and List B for $d \geq 5$ odd. It remains an open problem to search for other possible lists of priorities that improve the outcome of the algorithm. In Table 1, we get the asymptotic bisection width as solution to the differential equations (the tables reflect the constant to be multiplied by n). We may compare with the asymptotic lower bound of Bollobás and the asymptotic value of the upper bound of Alon, which is a deterministic one. For instance, for $d = 5$, Bollobás' lower bound yields $0.31917n$, Alon's upper bound yields $1.15118n$, while our upper bound is $0.5028n$. Furthermore, the complementary lower bounds we get on max bisection are well above the known lower bounds for all d-regular graphs, triangle-free or not.

Moreover, as can bee seen from Table 2, even for rather small values of n, the size of the bisection obtained by the algorithm is close to the solution determined by the differential equations. As n grows, this phenomenon strengthens.

The main open problem is to improve the upper bound on the bisection width. One way to do this may be to find better priority lists.

References

[1] N. Alon. On the edge-expansion of graphs. *Combinatorics, Probability and Computing*, (6):145–152, 1997.

[2] S.L. Bezroukov, R. Elsasser, B. Monien, R. Preis, and J.P. Tillich. New spectral lower bounds on the bisection width of graphs. In: U. Brandes and D. Wagner, editors, *26th. Graph-Theoretic Concepts in Computer Science*, volume 1928, pages 23–34. Springer, Lecture Notes in Computer Science, 2000.

[3] S. Boettcher, and A. Percus. Extremal optimization for graph partitioning. *Physical Review*, 64:26114-1–26114-13, 2001.

[4] B. Bollobás. The isoperimetric number of random regular graphs. *European Journal of combinatorics*, 9:241–244, 1984.

[5] B. Bollobás. *Random Graphs. Second Edition.* Combridge University Press, 2001.

[6] T. Bui, S. Chaudhuri, T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7:171–191, 1987.

[7] J. Díaz, N. Do, M.J. Serna, and N.C. Wormald. Bounds on the max and min bisection of random cubic and random 4-regular graphs. *Theoretical Computer Science*, 307:531–548, 2003.

[8] J. Díaz, J. Petit, and M.J. Serna. A survey on graph layout problems. *ACM Computing Surveys*, 34:313–356, 2002.

[9] W. Duckworth and N.C. Wormald Minimum independent dominating sets of random cubic graphs. *Random Structures and Algorithms*, 21:147–161, 2002.

[10] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. In: *41st. IEEE Annual Symposium on Foundation of Computer Science*, pages 23–26, 2000.

[11] U. Feige, M. Karpinski, and M. Langberg. A note on approximating Max Bisection on regular graphs. *Information Processing Letters*, (79):181–188, 2001.

[12] M. Fiedler. A property of the eigenvectors of nonnegtive symetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, (25):619–633, 1975.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[14] E. Halpering, D. Livnat, and U. Zwick. Max Cut in cubic graphs. In: *13th. Symposium on Discrete Algorithms*, pages 506–513, 2001.

[15] E. Halpering, and U. Zwick. A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems. *Random Structures and Algorithms*, (20):382–402, 2002.

[16] K. Jansen, M. Karpinski, A. Lingas, and E. Seidel. Polynomial time approximation schemes for MAX-BISECTION on planar and geometric graphs. In: A. Ferreira and H. Reichel, editors, *18th. Ann. Symposium on Theoretical Aspects of Computer Science*, volume 1938, pages 365–375. Springer, Lecture Notes in Computer Science, 2001.

[17] M. Jerrum, and G. Sorkin. The Metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 8:155–175, 1998.

[18] A.V. Kostochka and L.S. Melnikov. On bounds of the bisection width of cubic graphs. In: J. Nesetril and M. Fiedler, editors, *Fourth Czechoslovakian Symposium on Combinatorics, Graphs and Complexity*, pages 151–154. Elsevier Science Publishers, 1992.

[19] S.C. Locke. Maximum k-colorable subgraphs. *J. Graph Theory* 6(2):123–132, 1982.

[20] B. Monien and R. Preis. Upper bounds on the bisection width of 3 and 4-regular graphs. In: A. Pultr J. Sgall and P. Kolman, editors, *Mathematical Foundations of Computer Science*, volume 2136, pages 524–536. Springer, Lecture Notes in Computer Science, 2001.

[21] J.B. Shearer. A note on bipartite subgraphs of triangle-free graphs. *Random Structures Algorithms* 3(2):223–226, 1992.

[22] A. Steger and N.C. Wormald. Generating random regular graphs quickly. *Combinatorics, Probabability and Computing*, 8:377–396, 1999.

[23] N.C. Wormald. Differential equations for random processes and random graphs. *Annals of Applied Probability*, 5:1217–1235, 1995.

[24] N.C. Wormald. The differential equation method for random graph processes and greedy algorithms. In M. Karoński and H. Prömel, editors, *Lectures on Approximation and Randomized Algorithms*, pages 73–155. PWN, Warsaw, 1999.

[25] N.C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics*, pages 239–298. Cambridge University Press, 1999.

[26] N.C. Wormald. Analysis of greedy algorithms on graphs with bounded degree. EuroComb'01 (Barcelona). *Discrete Mathematics*, 273:235–260, 2003.