Uncover low degree vertices and minimise the mess: independent sets in random regular graphs

William Duckworth[1] and Michele Zito[2]

[1] Mathematical Sciences Institute
Australian National University
Canberra, ACT 0200, Australia.
e-mail: Billy.Duckworth@maths.anu.edu.au

[2] Department of Computer Science
University of Liverpool
Liverpool L69 3BX (UK).
e-mail: M.Zito@csc.liv.ac.uk

Abstract. We present algorithmic lower bounds on the size of the largest independent sets of vertices in a random d-regular graph. Our bounds hold with probability approaching one as the size of the graph tends to infinity.

1 Introduction

Given a graph $G = (V, E)$, an *independent set* is a subset U of V which spans no edge. In this paper we are interested in finding (by algorithmic means) independent sets of the largest possible cardinality. Let $\alpha(G)$ be the size of the largest independent sets in G divided by $|V(G)|$. The problem has a long history. It is one of the first optimisation problems whose decision version was shown to be NP-complete [12]. Since then many results have appeared either proving that an optimal structure can be found in polynomial time on special graph classes [1, 13] or showing that particular polynomial-time heuristics return solutions that are not too small for particular classes of graphs [5, 6, 8, 15] or else proving that finding heuristics returning solutions significantly close to the optimal ones is at least as hard as solving the optimisation problem exactly [14].

The algorithmic results in this paper are valid with probability approaching one as $|V(G)|$ tends to infinity (or *a.a.s.* in short), under the assumption that the input structure is presented according to a pre-specified probability distribution. The *maximum independent set problem* (MIS) has been studied thoroughly in several of such random structures. For graphs generated according to the well-known $G(n, p)$ model (n vertices, edges appear independently with probability p) it has been proven [7, 10] that as long as pn tends to infinity $\alpha(G(n,p))n \sim 2\log np/\log 1/(1-p)$ (although no polynomial time algorithm is known which returns, even just a.a.s., an independent set containing more than half that many vertices). For random d-regular graphs the situation is less satisfactory. For large d, $\alpha(G)$ is close to $\frac{2\log d}{d}$ [11]. However if d is a small fixed constant (say 3 or 5 or even 100), only lower and upper bounds are known. The best known bounds are reported in the second and third column of Table 1 for d up to 7. The upper bounds are derived using an argument based on Markov's inequality [17]. The lower bounds are algorithmic [18]. It is quite interesting that for the past 12 years, nobody has been able to improve these bounds (in fact the upper bounds are even older than that). More to this, the existence of greedy algorithms that return independent sets asymptotically larger than those promised by Wormald's algorithm, is an open problem raised ten years ago by Frieze and McDiarmid [9]. Notice that combinatorial methods like the one studied in [2], although successful for other packing problems [3, 4], failed to shed any light on the exact location of $\alpha(G)$ [2, Chap. IV].

In this paper we argue that careful algorithm design may be of significant help in improving the current situation. We propose a couple of heuristics that, when followed by an algorithm, lead to improvements on the performances of Wormald's algorithm for all values of d with only minimal additional running time costs. Wormald [18] showed that a simple process (termed "neighbourly" algorithm in that paper) that repeatedly picks vertices of minimum positive degree, adds them to the independent set that is being built and then removes all edges at distance at most one from the chosen vertex, builds fairly large independent sets a.a.s. if $G \in \mathcal{G}(n,d\text{-reg})$. It turns out that, in some cases, it is more convenient to add to the independent set one of the neighbours v of the initially chosen vertex u, rather than u itself. More precisely, we will choose v if such choice is guaranteed to create a number of low degree vertices (*sparsification* principle) or if it leads to the removal of very few edges (*minimal mess* principle). A detailed description of our algorithm is in Section 3.

We contend that our solution is simple to analyse: the proof of our results relies on a standard application of the differential equation method, spelled out in [19] (the method is general enough to allow the analysis of even more sophisticated algorithms). Furthermore, it seems plausible that similar principles may lead to improvements for other optimisation problems.

In Section 2 we present the model of random regular graphs that we use and a statement of our main result. In Section 3 we describe our new algorithm. The final part of the paper is devoted to the proof of our result: we first briefly describe the differential equation method, then we fill in all the details needed in the specific case.

Table 1. A.a.s. bounds on $\alpha(G)$ for random d-regular graphs.

d	l.b.	u.b.	α_d
3	0.4328	0.4554	0.4348
4	0.3901	0.4163	0.3921
5	0.3566	0.3844	0.3593
6	0.3296	0.3580	0.3330
7	0.3071	0.3357	0.3106

2 Model and main result

Let $\mathcal{G}(n,d\text{-reg})$ denote the uniform probability space of d-regular graphs on n vertices. Notation $G \in \mathcal{G}(n,d\text{-reg})$ will signify that the graph G is selected according to such model.

A construction that gives the elements of $\mathcal{G}(n,d\text{-reg})$ is the *configuration model* (see, for example, [16, Chapter 9]). Let n urns be given, each containing d balls. A set F of $dn/2$ unordered pairs of balls is chosen u.a.r.. Let Ω be the set of all such pairings. Each $F \in \Omega$ corresponds to an d-regular (multi)graph with vertex set $V = \{1, \ldots, n\}$ and edge set E formed by those sets $\{i, j\}$ for which there is at least one pair with one ball belonging to urn i and the other ball belonging to urn j. Let Ω^* be the set of all pairings not containing an edge joining balls from the same urn or two edges joining the same two urns. Since each simple graph corresponds to exactly $(d!)^n$ such pairings, a random pairing $F \in \Omega^*$ corresponds to an d-regular graph G without loops or multiple edges chosen u.a.r.

Notice that a random pairing can be picked by choosing pairs one after the other. Moreover, the first point in a pair may be selected using any rule whatsoever, as long as the second point is chosen u.a.r. from all the remaining free (unpaired) points. This property implies the existence of two equivalent ways of describing the algorithm presented in this paper. The description in Section 3 works on a previously generated (random) graph. However it would be equally easy to define a process that, by working on configurations, at the same time generates the graph and simulates our algorithm (this approach might help understanding our analysis).

In this paper we prove the following result:

Theorem 1. *For every integer n and $d \geq 3$, if $G \in \mathcal{G}(n,d\text{-reg})$ then $\alpha(G) \geq \alpha_d$ a.a.s. where the values of α_d are obtained through the method described in Section 4 and are reported, for $d \leq 7$ in the fourth column of Table 1.*

The proofs of Theorem 1 is based on the definition of an algorithm and on the fact that, for each constant value of d, the algorithm dynamics can be described with sufficient precision by a random process that, for large n, behaves in a predictable way.

3 Greedy algorithms for independent sets

If $G = (V, E)$ is a graph then $\Gamma(v) = \{u \in G : \{u, v\} \in E(G)\}$, for each $v \in V(G)$. The degree of a vertex v of G is the size of $\Gamma(v)$. Let G be a graph whose maximum degree is bounded by some fixed constant $d > 0$. We call the *isolation of* u the process of deleting all edges at distance at most one from vertex u. For each $q \in \{1, \ldots, d\}$ define Op_q to be the process of picking uniformly at random a vertex u of degree q in G, and isolating it unless the minimal degree x in $\Gamma(u)$ is at most q, there is a vertex v of degree x in $\Gamma(u)$ and one of the following conditions hold (in such case v gets isolated):

1. there are at least two vertices of degree x in $\Gamma(u)$, OR
2. there is a single vertex v of degree x in $\Gamma(u)$ AND
   (a) $q = 2$ and the minimum degree in $\Gamma(v) \setminus u$ is larger than that in $\Gamma(u) \setminus v$ OR
   (b) $2 < q < d-1$, the minimum degree in $\Gamma(v) \setminus u$ is larger than q, and the sum of all degree in $\Gamma(v) \setminus u$ is smaller than that in $\Gamma(u) \setminus v$.

For each $q \in \{1, \ldots, d\}$ Op_q obeys the sparsification (cases 1. and 2.(a)) and minimal mess principles (case 2.(b)) described in Section 1. We may then consider the following process:

Careful minimum degree process. While there are still edges in G, define q as the minimum positive degree in G and perform Op_q on G, adding to $\mathcal{I}$ the vertex that has been (deliberately) isolated in the process.

If there is no edge left then return $\mathcal{I}$ and stop.

Note that other vertices (apart from u or v) may get isolated while performing Op_q. They are not taken into account by our analysis. Denote by G_t, for each integer t, the graph obtained from G by removing all edges at distance at most one from any of the first t vertices added to $\mathcal{I}$ (of course $G_0 \equiv G$).

The differential equation method [19] will allow us to estimate the size of $\mathcal{I}$ at the end of the process from (approximate) knowledge of the dynamics of the vector $(|V_1|, \ldots, |V_d|)$ (where $V_i = V_i(t) = \{v \in G_t : |\Gamma(v)| = i\}$, for each $i \in \{1, \ldots, d\}$). However, to avoid certain technicalities in the analysis, it will be convenient to analyse the following related process (here d is a fixed integer and ϵ a small positive real number):

```
Algorithm CarefulGreedy$_{d,\epsilon}(G)$
Input: a graph $G = (V, E)$ on $n$ vertices and maximum degree $d$.
    $\mathcal{I} \leftarrow \emptyset$;
    for $t \leftarrow 0$ to $\lceil \epsilon n \rceil$ perform Op$_d$;
    while $E \neq \emptyset$
        compute a probability distribution $p(q, \frac{t}{n}, \frac{|V_1|}{n}, \ldots, \frac{|V_d|}{n})$,
            for $q \in \{1, \ldots, d\}$;
        choose $q \in \{1, \ldots, d\}$ with probability $p(q, \frac{t}{n}, \frac{|V_1|}{n}, \ldots, \frac{|V_d|}{n})$;
        perform Op$_q$ on $G_t$;
        $t \leftarrow t + 1$;
    return $\mathcal{I}$.
```

A *step* of this algorithm is a complete iteration of the algorithm main while loop. Assuming that each vertex adjacencies are retrievable in time $O(d)$ and that all vertex degrees are computed before the main loop is executed and then updated as edges get removed, it is easy to believe that the algorithm time complexity is $O(dn)$.

Initially $V_d = V$ and $V_i = \emptyset$ for $i \leq d-1$. For $t > \lceil \epsilon n \rceil$, the choice to perform Op$_q$, for $q \in \{1, \ldots, d-1\}$ is based on a probability distribution $p(q, x, \mathbf{y})$. The general definition of $p(q, x, \mathbf{y})$, valid when G is a random d-regular graph, will be given in Section 4. Depending on the particular probability distribution $p(q, x, \mathbf{y})$ that is used at a given step, the algorithm will be in one of a number of different *phases*. The outcome of our analysis implies that the algorithm processing goes through successive phases. In phase $j \in \{1, 2, \ldots\}$ the process performs only Op$_{d-j}$ or Op$_{d-j-1}$. In this sense algorithm CarefulGreedy$_{d,\epsilon}$ simulates the careful minimum degree process described above.

4 Analysis method

In order to obtain estimates on the size of the independent set returned by algorithm CarefulGreedy$_{d,\epsilon}(G)$ we use the differential equation method proposed by Wormald [19]. Given the input graph, our algorithm peels off a number of edges (upper bounded by an expression depending only on d) from the graph G_t and updates the structure $\mathcal{I}$ ($\mathcal{I}_t$ will denote the content of $\mathcal{I}$ before G_t is further processed) that is being built. Let $Y_i(t) = |V_i(t)|$ for $i \in \{1, \ldots, d\}$ and $Y_{d+1}(t) = |\mathcal{I}_t|$. In what follows, for $i \in \{1, \ldots, d+1\}$ and $q \in \{1, \ldots, d-1\}$, functions $f_{i,q}$ in $\mathbb{R}^{d+2}$ will be such that the expected change to $Y_i(t)$, conditioned on the history of the process up to step t and following one occurrence of Op$_q$ during step $t+1$ is asymptotically $f_{i,q}(\frac{t}{n}, \frac{Y_1(t)}{n}, \ldots, \frac{Y_{d+1}(t)}{n}) + o(1)$, whenever $Y_q(t) > 0$. Assuming that these functions are continuous and bounded in

$$\mathcal{D}_\epsilon = \{(x, y_1, \ldots, y_{d+1}) : 0 \leq x \leq d,\ 0 \leq y_i \leq d \text{ for } 1 \leq i \leq d+1,\ y_d \geq \epsilon\}$$

we may consider the following $d-1$ distinct systems of differential equations

$$\frac{\mathrm{d}y_i}{\mathrm{d}x} = \frac{f_{d-s-1,d-s}(x,\mathbf{y})}{f_{d-s-1,d-s}(x,\mathbf{y})-f_{d-s-1,d-s-1}(x,\mathbf{y})} f_{i,d-s-1}(x,\mathbf{y}) + \\ - \frac{f_{d-s-1,d-s-1}(x,\mathbf{y})}{f_{d-s-1,d-s}(x,\mathbf{y})-f_{d-s-1,d-s-1}(x,\mathbf{y})} f_{i,d-s}(x,\mathbf{y}) \quad (1)$$

for $s \in \{1,\ldots,d-2\}$, and also, $\frac{\mathrm{d}y_i}{\mathrm{d}x} = f_{i,1}(x,\mathbf{y})$. If the functions $f_{i,q}$ are rational with no pole in $\mathcal{D}_\epsilon$ and there exist positive constants C_1, C_2, and C_3 such that for each $i \in \{1,\ldots,d\}$, everywhere in $\mathcal{D}_\epsilon$, $f_{i,q} \geq C_1 y_{i+1} - C_2 y_i$ (for $q \neq i$), and $f_{i,q} \leq C_3 y_{i+1}$ for all q (see [19]) then each of the systems in (1), coupled with a suitably defined initial condition, admits a unique solution over an $[x_{s-1}, x_s]$ (for $s \in \{1,\ldots,d-1\}$), where

$x_0 = 0$ and x_s is defined as the infimum of those $x > x_{s-1}$ for which at least one of the following holds:

(C1) $f_{d-s-1,d-s-1}(x,\mathbf{y}) \geq 0$ or
$f_{d-s-1,d-s}(x,\mathbf{y}) - f_{d-s-1,d-s-1}(x,\mathbf{y}) \leq \epsilon$ and $s < d-1$;

(C2) the component $d-s$ of the solution falls below zero or

(C3) the solution is outside $\mathcal{D}_\epsilon$ or ceases to exist. (2)

Let $\tilde{\mathbf{y}} = \tilde{\mathbf{y}}(x) = (\tilde{y}_1(x),\ldots,\tilde{y}_{d+1}(x))$ be the function defined inductively as follows:

For each $i \in \{1,\ldots,d+1\}$, $\tilde{y}_i(0) = \frac{Y_i(0)}{n}$. For $s \geq 1$, $\tilde{\mathbf{y}}$ is the solution to (1) over $[x_{s-1}, x_s]$, with initial condition $\mathbf{y}(x_{s-1}) = \tilde{\mathbf{y}}(x_{s-1})$. (3)

We may now state the result which bounds from below $\alpha(G)$. The values of m and $\tilde{y}_{d+1}(x_m)(=\alpha_d)$ referred to in Theorem 2 were found solving the various systems numerically using `Maple`'s Runge-Kutta Fehlberg method (a very primitive solver written in `C` for the case $d = 3$ is enclosed in the Appendix). The distributions used in algorithm CarefulGreedy$_{d,\epsilon}(G)$ satisfy the following definition

$$p(q,x,\mathbf{y}) = \begin{cases} -\frac{f_{d-s-1,d-s-1}(x,\mathbf{y})}{f_{d-s-1,d-s}(x,\mathbf{y})-f_{d-s-1,d-s-1}(x,\mathbf{y})} & q = d-s \\ \frac{f_{d-s-1,d-s}(x,\mathbf{y})}{f_{d-s-1,d-s}(x,\mathbf{y})-f_{d-s-1,d-s-1}(x,\mathbf{y})} & q = d-s-1 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

when $x \in [x_{s-1}, x_s]$, for each $s \in \{1,\ldots,m\}$.

Theorem 2. *Let d be a positive integer with $d \geq 3$, and ϵ an arbitrarily small positive real number. For $q \in \{1, \ldots, d-1\}$, let $f_{i,q}$, for each $i \in \{1, \ldots, d+1\}$ be the functions referred to in the description above and defined in Sections 4.1 and 4.2. Then there exists a positive integer m such that the algorithm CarefulGreedy$_{d,\epsilon}(G)$ a.a.s. returns a structure of size $n\tilde{y}_{d+1}(x_m) + o(n)$ where functions $\tilde{y}_1, \ldots, \tilde{y}_{d+1}$ are defined in (3) and $x_0, \ldots, x_m$ in (2) when $G \in \mathcal{G}(n,d$*-reg).

The proof of this result is carried out invoking Theorem 1 in [19]. The important point to stress is that the argument has two quite separate components. The definition of a number of functions and numerical quantities (satisfying certain conditions) related to the particular algorithm and the proof that everything works and the solutions of (1) actually give us information on $|\mathcal{I}|$ after the execution of CarefulGreedy$_{d,\epsilon}(G)$. As long as we are able to define the various $f_{i,q}$ and verify that various smoothness conditions are satisfied, we do not need to be concerned with the second part of the argument (which will mirror Wormald's general argument). The quantitative analysis of CarefulGreedy$_{d,\epsilon}(G)$ is thus reduced to the definition of a number of functions and parameters that directly relate to the algorithm processing.

Before digging into the details of the specific cases we introduce few notations specific to $\mathcal{G}(n,d$-reg). In what follows for integers a and b, $\delta_{a,b}$ is equal to one (resp. zero) if $a = b$ (resp. otherwise). Given a vertex u, the probability of creating a vertex of degree $i-1$ in $\Gamma(u)$ when removing an edge incident to u is asymptotically $P_i = \frac{iY_i}{\sum iY_i}$. In what follows S_a^b will denote the sum of all P_i's for $a \leq i \leq b$ (with $S_a^b = 0$ if $a > b$). Furthermore let $\text{Min}_c(a) = (S_a^b)^c - (S_{a+1}^b)^c$. For large n, $\text{Min}_c(a)$ approximates the probability that the minimum degree in a given set of c vertices is a, given that all degrees are between a and b. The expected change in Y_i due to the degree changes in $\Gamma(u)$ following the removal of an edge incident to u can be approximated by $\rho_i = P_{i+1} - P_i$ with $P_{d+1} = 0$. Similarly, if $e = \{u, v\}$ the expected change in Y_i due to the removal of e and of any other edge incident to v is asymptotically $\mu_i = -P_i + \rho_i \sum_{z=2}^{d} P_z(z-1)$. Finally, if $\mathcal{P}$ is some boolean condition, define $\diamond_r(\mathcal{P}) = (r-1)\rho_i - \delta_{i,r}$ (resp. $\delta_{i,r-1} - \delta_{i,r}$) if $\mathcal{P}$ is true (resp. false).

4.1 The simple case $d = 3$

Before describing the general case, it may be useful to follow an informal description of the analysis on cubic graphs.

For $d = 3$, we may assume that, at the beginning of a step, vertex u has degree either one or two (the initial $\lceil \epsilon n \rceil$ steps will make sure that this assumption is valid). Algorithm CarefulGreedy$_{3,\epsilon}(G)$ behaves exactly like Wormald's algorithm except in the case when we perform Op_2 and the two neighbours of u both have degree two. In such case our algorithm chooses a random neighbour of u rather than u itself. Thus, if $f^W_{i,q}$ denotes the function $f_{i,q}$ associated with Wormald's algorithm (a precise definition is given in formula (2.12) of [19]) then $f_{i,q} = f^W_{i,q} + \delta_{q,2}(P_2)^2(\delta_{i,1} + \mu_i - 2\rho_i)$, for $i \in \{0, \dots 3\}$ and $q \in \{1, 2\}$ (whereas $f_{4,q} = f^W_{4,q}$). Of course each $f_{i,q}$ satisfies the conditions that imply Theorem 2 (this follows from the properties of $f^W_{i,q}$ in [19]). For $d = 3$ it turns out that $m = 1$. Conditions (C2) eventually becomes true exactly when the vector (x, y_1, y_2, y_3, y_4) hits the boundary of $\mathcal{D}_\epsilon$. At that point the process stops and $\tilde{y}_4 \simeq 0.4347531298$, which agrees with the value for α_3 stated in Table 1.

4.2 Arbitrary $d \geq 4$

We next state the general result characterising the dynamics of $(Y_1, \dots, Y_{d+1})$ for arbitrary $d \geq 4$ following an instance of Op_q, for $q \in \{1, \dots, d-1\}$.

Lemma 1. *Let $d \geq 4$ and $\epsilon > 0$. For each $q \in \{1, \dots, d-1\}$, conditioned on the history of algorithm* CarefulGreedy$_{d,\epsilon}(G)$ *up to step t, the expected change to $Y_i(t)$ following one occurrence of* Op_q *is asymptotically*

$$
\begin{aligned}
-\delta_{i,q} \quad &+ (S^d_{q+1})^q \quad \times \textstyle\sum_{k=q+1}^d q \frac{P_k}{S^d_{q+1}}((k-1)\rho_i - \delta_{i,k}) + \\
&+ \textstyle\sum_{x=1}^q \quad \Big[(\mathrm{Min}_q(x) - qP_x(S^d_{x+1})^{q-1})\,((x-1)\mu_i - \delta_{i,x}) + \\
&\qquad - (qP_x(S^d_x)^{q-1} - \mathrm{Min}_q(x))\,(\delta_{i,x} - \delta_{i,x-1}) + \\
&\qquad - q(\mathrm{Min}_{q-1}(x) - (q-1)P_x(S^d_{x+1})^{q-2}) \textstyle\sum_{k=x+1}^d P_k(\delta_{i,k} - \delta_{i,k-1})\Big] + \\
&+ qP_1(S^d_2)^{q-1}(-\delta_{i,1} + (q-1)\textstyle\sum_{k=2}^d((k-1)\rho_i - \delta_{i,k})\frac{P_k}{S^d_2}) + \\
&+ \textstyle\sum_{x=2}^q qP_x\Big\{\sum_{z=x+1}^{d-1}\Big[\ \sum_{m=1}^{q+(d-q)\delta_{q,2}}\Big[-\delta_{i,x}\mathrm{Min}_{q-1}(z)\mathrm{Min}_{x-1}(m) + \\
&+ (q-1)\mathrm{Min}_{x-1}(m) \times ((S^d_z)^{q-2} \diamond_z (m \leq z)P_z + \mathrm{Min}_{q-2}(z)\textstyle\sum_{r=z+1}^d \diamond_r(m \leq z)P_r) + \\
&+ (x-1)\mathrm{Min}_{q-1}(z) \cdot ((S^d_m)^{x-2} \diamond_m (m > z)P_m + \mathrm{Min}_{x-2}(m)\textstyle\sum_{s=m+1}^d \diamond_s(m > z)P_s)\Big] + \\
&+ \textstyle\sum_{m=q+(d-q)\delta_{q,2}+1}^d \sum_{\boldsymbol{j}:j_z>0}\binom{q-1}{j_z,\dots,j_d}P_z^{j_z}\dots P_d^{j_d}\sum_{\boldsymbol{k}:k_m>0}\binom{x-1}{k_m,\dots,k_d}P_m^{k_m}\dots P_d^{k_d}\gamma(\boldsymbol{j},\boldsymbol{k})\Big] + \\
&\qquad + P_d^{q-1}((x-1)\rho_i - \delta_{i,x} + ((d-1)\rho_i - \delta_{i,d})(q-1))\Big\}.
\end{aligned}
$$

where $\gamma(\boldsymbol{j}, \boldsymbol{k}) = \sum_{r=z}^d \diamond_r(\mathcal{P}) \cdot j_r - \delta_{i,x} - \sum_{r=m}^d \diamond_r(\neg\mathcal{P}) \cdot k_r$ and $\mathcal{P} \equiv \sum_{r=z}^d rj_r < \sum_{r=m}^d rk_r$, if $i \leq d$. Finally $f_{d+1,q} = 1$ for all values of q.

Remark. The first line in the asymptotic expression for $f_{i,q}$ ($i \le d$) refers to the case when the minimum degree around u is at least $q+1$. The subsequent sum deals with the case when there is at least two vertices of minimum degree $x \le q$ in $\mathit{\Gamma}(u)$. The remainder of the expression covers the case when there is a single vertex of minimum degree $x \le q$ in $\mathit{\Gamma}(u)$.

Proof. We sketch the definition of $f_{i,q}$ for each q in the given range and $i \le d$. The stated expression (more convenient from the numerical point of view) can then be obtained through simple algebraic manipulations.

For arbitrary, fixed $d \ge 4$, each iteration in the while loop of algorithm CarefulGreedy$_{d,\epsilon}(G)$ may perform Op$_q$ for $q \in \{1,\dots,d-1\}$. More importantly the execution of such an operation generates a number of alternative events whose probabilities can be approximated quite easily under the assumption that $G \in \mathcal{G}(n, d\text{-reg})$. Hence, for each $i \in \{1,\dots,d\}$ and $q \in \{1,\dots,d-1\}$ function $f_{i,q}$ satisfies (a.a.s.) the following definition:

$$f_{i,q} = \; -\delta_{i,q} + \sum \tbinom{q}{j_{q+1},\dots,j_d} P_{q+1}^{j_{q+1}} \dots P_d^{j_d} \left(\textstyle\sum_{k=q+1}^{d} ((k-1)\rho_i - \delta_{i,k}) j_k\right) +$$

$$\textstyle\sum_{x=1}^{q} \{ \quad g_{i,q,x} \; + \; \sum_{\boldsymbol{j}: j_x > 1} \tbinom{q}{j_x,\dots,j_d} P_x^{j_x} \dots P_d^{j_d} ((x-1)\mu_i - \delta_{i,x} +$$

$$- \textstyle\sum_{k=x}^{d} (\delta_{i,k} - \delta_{i,k-1})(j_k - \delta_{k,x}) \}.$$

where the first sum is over all sequences of non-negative integers $(j_{q+1},\dots,j_d)$ adding up to q, the second sum on the second line is over all $(j_x,\dots,j_d)$ with the further restriction that j_x must be positive (x represents the minimum degree in $\mathit{\Gamma}(u)$), and $g_{i,q,x}$ is the expected change to Y_i conditioned on performing Op$_q$ and on the existence of a single vertex of minimum degree x around u. Function $g_{i,d-1,x}$ has a very simple definition, since if we perform Op$_{d-1}$ we are essentially just replicating Wormald's algorithm. To define $g_{i,q,x}$ for $q < d-1$ we need to condition on the degree structure in $\mathit{\Gamma}(v) \setminus u$ and $\mathit{\Gamma}(u) \setminus v$. In particular if $x = 1$ then $\mathit{\Gamma}(v) \setminus u$ is empty and therefore (just following Wormald's algorithm) $g_{i,q,1} = \sum \tbinom{q}{1,j_s,\dots,j_d} P_1 \dots P_d^{j_d} (-\delta_{i,1} + \sum_{k=2}^{d} ((k-1)\rho_i - \delta_{i,k}) j_k)$. For $x \ge 2$,

$$g_{i,q,x} = q P_x \Big\{ \textstyle\sum_{z=x+1}^{d-1} \Big[h_{i,q,x,z} + \sum_{m \ne z} \sum_{\boldsymbol{j}: j_z > 0} \tbinom{q-1}{j_z,\dots,j_d} P_z^{j_z} \dots P_d^{j_d} \times$$

$$\times \textstyle\sum_{\boldsymbol{k}, k_m > 0} \tbinom{x-1}{k_m,\dots,k_d} P_m^{k_m} \dots P_d^{k_d} \gamma_{i,q,x,z,m} \Big] +$$

$$+ \quad P_d^{q-1} ((x-1)\rho_i - \delta_{i,x} + ((d-1)\rho_i - \delta_{i,d})(q-1)) \Big\}$$

where $h_{i,q,x,z}$ describes the case when the minimum degree in $\mathit{\Gamma}(u) \setminus v$ and $\mathit{\Gamma}(v) \setminus u$ are the same and $\gamma_{i,q,x,z,m}$ the expected updates necessary in any other case. If $z \neq m$ the algorithm's rule is quite simple: v is added to $\mathcal{I}$ if the minimum degree in $\Gamma(v) \setminus u$ is larger than that in $\Gamma(u) \setminus v$. Then $\gamma_{i,q,x,z,m} = \sum_{r=z}^{d} \diamond_r(m \leq z)j_r + \sum_{s=m}^{d} \diamond_s(m > z)k_s$. Finally, for $q > 2$, function $h_{i,q,x,z}$ is computed conditioning on each possible pair of sequences $(j_z, \ldots, j_d)$ and $(k_z, \ldots, k_d)$ adding up to $q-1$ and $x-1$ respectively, and having $j_z > 0$ and $k_z > 0$. Vertex v is added to $\mathcal{I}$ if $\sum_{r=z}^{d} rj_r > \sum_{r=m}^{d} rk_r$. $\square$

References

1. V. E. Alekseev. Polynomial algorithm for finding the largest independent sets in graphs without forks. *Discrete Applied Mathematics*, 135(1–3):3–16, 2004.
2. H. Assiyatun. *Large Subgraphs of Regular Graphs*. PhD thesis, Department of Mathematics and Statistics - The University of Melbourne, 2002.
3. H. Assiyatun and N. Wormald. 3-star factors in random d-regular graphs. *European Journal of Combinatorics*, 27(8):1249–1262, 2006.
4. Hilda Assiyatun. Maximum induced matchings of random regular graphs. In *Combinatorial geometry and graph theory*, volume 3330 of *Lecture Notes in Comput. Sci.*, pages 44–57. Springer, Berlin, 2005.
5. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the Association for Computing Machinery*, 41(1):153–180, January 1994.
6. P. Berman and T. Fujito. On approximation properties of the independent set problem for low degree graphs. *Theory of Computing Systems*, 32(2):115–132, 1999.
7. B. Bollobás and P. Erdős. Cliques in random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 80:419–427, 1976.
8. Z-Z. Chen. Approximation algorithms for independent sets in map graphs. *Journal of Algorithms*, 41(1):20–40, 2001.
9. A. Frieze and C. McDiarmid. Algorithmic theory of random graphs. *Random Structures and Algorithms*, 10:5–42, 1997.
10. A. M. Frieze. On the independence number of random graphs. *Discrete Mathematics*, 81(2):171–175, 1990.
11. A. M. Frieze and T. Łuczak. On the independence and chromatic number of random regular graphs. *Journal of Combinatorial Theory*, B 54:123–132, 1992.
12. M. R. Garey and D. S. Johnson. *Computer and Intractability, a Guide to the Theory of NP-Completeness*. Freeman and Company, 1979.
13. F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, June 1972.
14. J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathem.*, 182:105–142, 1999.
15. H. B. Hunt, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation scheme for NP- and PSPACE-hard problems for geometric graphs. *Journal of Algorithms*, 26:238–274, 1998.
16. S. Janson, T. Łuczak, and A. Ruciński. *Random Graphs*. John Wiley and Sons, 2000.
17. B. D. McKay. Independent sets in regular graphs of high girth. *Ars Combinatoria*, 23A:179–185, 1987.
18. N. C. Wormald. Differential equations for random processes and random graphs. *Annals of Applied Probability*, 5:1217–1235, 1995.
19. N. C. Wormald. Analysis of greedy algorithms on graphs with bounded degrees. *Discrete Mathematics*, 273:235–260, 2003.

Appendix

We enclose a short C program that solves the single system relevant for $d = 3$. The final print statement outputs α_3. While the code is by no mean numerically sound (in particular it contains no checking on $f_{i,q}$ or $\tilde{y}$) the fact that it gives an answer that is very close to the one returned by Maple's solver may be taken as further evidence of the robustness of our numerical results.

```
#include <stdio.h>

inline double pw(double value, int pwer) {
  double answer=1.0;
  int i;
  for (i=pwer;i;i--) answer=answer*value;
  return answer;
}

double p (int i, double y[]) {return i*y[i]/(y[1]+2*y[2]+3*y[3]);}
double R (int i, double y[]) {return (i<3?p(i+1,y):0.0) - p(i,y);}
double M (int i, double y[]) {return -p(i,y)+R(i,y)*(p(2,y)+2*p(3,y));}

double f (int i, int q, double y[]) {
  if (i==4) return 1.0;
  else return -(i==q)+q*M(i,y)-(q==2)*pw(p(2,y),2)*(2*R(i,y)-M(i,y)-(i==1));
}

double F(int i,double y[]) {
  double p2=-f(1,1,y)/(f(1,2,y)-f(1,1,y));
  return p2*f(i,2,y)+(1-p2)*f(i,1,y);
}

main(int argc, char *argv[]) {
  int i,l;
  double h = 0.00000001;
  double w[5],mid[5];

  for(i=0;i<5;i++) w[i]=0.0+(!(3-i));

  for (l=0;;l++) {
    for (i=0;i<5;i++) mid[i]=w[i]+(h*F(i,w)/2.0);
    for (i=0;i<5;i++) w[i]=w[i]+h*F(i,mid);

    if ( (f(1,1,w)>0||f(1,2,w)-f(1,1,w)<=h) || (w[2]<=0.0) ) {
      printf("|I| = %11.10f\n",w[4]); break;
    }
  }
}
```