Maximum 2-independent sets of random cubic graphs

W. Duckworth*

Department of Computing
Macquarie University
Sydney, NSW 2019
Australia
billy@ics.mq.edu.au

Abstract

We present a simple, yet efficient, heuristic for finding a large 2-independent set of cubic graphs. We analyse the performance of this heuristic, which is a randomised greedy algorithm, on random n-vertex cubic graphs using differential equations. In this way, we are able to prove that the expected size of the 2-independent set returned by the algorithm is asymptotically almost surely greater than $0.20485n$.

1 Introduction

A *k-independent set* of a graph, G, is a subset, $\mathcal{I}$, of the vertices of G such that the distance between any two vertices of $\mathcal{I}$ in G is at least $k + 1$. We are interested in finding k-independent sets of large cardinality. Kong and Zhao [7] showed that for every $k \geq 2$, finding a *maximum* k-independent set of a graph is NP-hard, even when restricted to regular bipartite graphs [8]. For other basic graph theory definitions not defined here, the reader is referred to, for example, Diestel [3].

Finding a large k-independent set of a graph has applications in the fields of job-scheduling on k-machines, VLSI design layout, routing and channel assignment location [5]. Many of these applications are in the field of distributed computing [9] and, as networks often have bounded or even regular degree, it is of interest to consider algorithms for finding large k-independent sets of such graphs.

Many problems that remain NP-hard when the input is restricted to d-regular graphs are often polynomial time solvable when $d \leq 2$. Therefore, cubic (i.e. 3-regular) graphs may be considered to be of most interest in this regard.

* This research was carried out whilst the author was in The Department of Mathematics and Statistics, The University of Melbourne, VIC 3010, Australia

Due to the NP-hardness of the k-independent set problem, we are forced to relax the optimality requirement and consider heuristics that find a solution that is somehow close to optimal in a time that is bounded by a polynomial of the input size. Simple heuristics often have a relatively poor worst-case performance as there may exist many extremal input instances on which a simple algorithm may perform badly. It is therefore natural to consider the average-case performance of such heuristics. The performance of a simple heuristic may still be poor on average and therefore more efficient algorithms may need to be developed. However, for $k > 2$ (in particular for $k \geq 4$) it seems difficult to find efficient heuristics that find large k-independent sets of (regular) graphs that have a reasonable average-case performance, let alone, worst-case performance. The heuristic for the 2-independent set problem we present does not readily extend to larger values of k (and not at all for values of $k > 3$). Heuristics that we have analysed that extend to larger values of k have a poorer performance than the algorithm we present for $k = 2$.

We consider simple, connected, cubic graphs. When considering any such graph on n vertices, we assume n to be even to avoid parity problems. Note that for n-vertex cubic graphs, it is simple to show that the size of a maximum 2-independent set is at most $n/4$ and at least $n/10$ implying that this problem is trivially approximable with approximation ratio 2.5 for such graphs.

In [4] a deterministic algorithm for finding a large 2-independent set of cubic graphs was presented and analysed. It was shown that the size of a maximum 2-independent set of an n-vertex cubic graph is at least $n/8 + O(1)$. The linear programming technique that was used to analyse the performance of the algorithm that was presented, also demonstrated the existence of an infinite family of cubic graphs for which the algorithm only achieves this bound.

As we consider cubic graphs that are generated u.a.r. (uniformly at random), we need some notation. We use the notation $\mathbf{P}$ (probability), $\mathbf{E}$ (expectation) and say that a property, $\mathcal{B} = \mathcal{B}_n$, of a random regular graph on n vertices holds a.a.s. (asymptotically almost surely) if $\lim_{n\to\infty} \mathbf{P}(\mathcal{B}_n)$=1. For other random graph theory definitions not defined here, the reader is referred to, for example, Janson *et al.* [6].

Assiyatun [1] recently gave existence proofs that bound the size of a maximum 2-independent set of random regular graphs. These results imply that the size of a maximum 2-independent set, I, of a random n-vertex cubic graph a.a.s. satisfies $0.2315n \leq |I| \leq 0.2356n$.

We present a simple, yet efficient, heuristic for finding a large 2-independent set of cubic graphs. We analyse the performance of this heuristic, which is a randomised algorithm, on random n-vertex cubic graphs using differential equations. We prove that the expected size of the 2-independent set returned by the algorithm is a.a.s. greater than $0.20485n$.

The following section gives a brief description of our algorithm along with some motivation as to why this algorithm has a reasonable average-case performance. In Section 3 we describe the model we use for generating cubic graphs u.a.r. and describe the notion of analysing the performance of algorithms on random graphs using systems of differential equations. Details of our algorithm are given in Section 4 and its analysis is presented in Section 5 proving our a.a. sure lower bound.

2 A Greedy Heuristic

One of the simplest types of algorithm that may be used to approximate an NP-hard graph-theoretic optimisation problem are *greedy* algorithms. These algorithms iteratively make choices that seem (at the time) to be the most beneficial. At each iteration, as each subsequent member of the set under construction is chosen, the algorithm updates the graph by removing vertices and (or) edges in order that, once the next selection has been made, the set constructed thus far satisfies the given requirements (independence etc.).

At first glance, the simplest heuristic for approximating the 2-independent set problem would be to repeatedly choose a vertex, v, for inclusion in the set, each time, deleting all vertices at distance at most 2 from v (any such vertex is within distance 3 of v and would therefore not be allowed to be chosen as part of the set, having already chosen v). Unfortunately this heuristic is not guaranteed to produce a set that is 2-independent. Consider a subgraph of the input graph with vertex set $\{1, 2, 3, 4, 5, 6, 7\}$ and edge set $\{(1, 2), (2, 3), (3, 4), (3, 5), (4, 6), (5, 7)\}$. Choose vertex 1 to be part of the 2-independent set and delete all vertices at distance at most 2 from vertex 1. This leaves the edges $(4, 6)$ and $(5, 7)$ intact. Then choose vertex 4 (which is at distance 3 from vertex 1) to be part of the set. The algorithm deletes vertices 4 and 6 but no more as it has no knowledge that vertices 4 and 5 *were* connected by a path of length 2. It could then continue and pick vertex 5 (which is at distance 2 from vertex 4) to be part of the set.

A variation on this (for a cubic graph) would be to repeatedly select vertices of degree 3 for inclusion in the set. Each time a vertex is chosen, delete all of its neighbours. This is guaranteed to return a set that is 2-independent. However, this algorithm has a worse average-case performance than the algorithm we present.

The heuristic we describe is a randomised greedy algorithm that is based on repeatedly selecting vertices of given current degree from an ever-shrinking subgraph of the input graph. At the start of our algorithm all vertices have degree 3. Throughout the algorithm, as vertices are chosen for inclusion in the set under construction, edges are deleted and the algorithm terminates when all vertices have degree 0.

For a cubic graph, G, the algorithm constructs a subset, $\mathcal{I}$, of the vertices of G in a series of *steps*. Each step starts by selecting a vertex u.a.r. from those vertices of a particular current degree. The first step is unique in the sense that it is the only step in which a vertex is selected u.a.r. from the vertices of degree 3. We select such a vertex, u, u.a.r. from all the vertices of the input graph to add to $\mathcal{I}$. We then delete all edges incident with u and its neighbours. Note that, as we assume the input graph to be connected, after the first step and before the completion of the algorithm, there always exists a vertex of current degree 1 or 2.

For each step after the first, if there exist vertices of current degree 1, such a vertex, u, is chosen u.a.r. Otherwise we select u u.a.r. from those vertices of current degree 2. We then investigate the degree(s) of the neighbour(s) of u. If u has one or more neighbours of degree 3, we select such a vertex, v, u.a.r. to add to $\mathcal{I}$ and delete all edges incident with v and its neighbours. If u has no neighbour of degree 3, we delete the edges incident with u (without adding a vertex to $\mathcal{I}$).

Note that each vertex chosen to be part of $\mathcal{I}$ is chosen from the vertices of degree 3 in the step that it is selected. Once each selection has been made, all edges incident with the chosen vertex and its neighbours are deleted. These two facts guarantee that the set $\mathcal{I}$ returned by the algorithm is 2-independent in G.

The steps in which the algorithm does not add a vertex to $\mathcal{I}$ are those when the chosen vertex, v, and all its neighbours have degree less than 3. In such an instance, choosing v to be part of $\mathcal{I}$ *may* lead to the set returned not being 2-independent in G. As v had current degree less than 3, there must have been at least one edge incident with v that has been deleted in a previous step. This edge may form part of a path of length 2 between v and another vertex that has already been selected (or that may be selected at some later time).

3 Random Graphs and Differential Equations

3.1 Generating Random Cubic Graphs

The model we use to generate a cubic graph u.a.r. was first described in its simplest form by Bollobás [2] and may be summarised as follows. For an n-vertex cubic graph: take $3n$ points in n buckets labelled $1 \ldots n$ (with three points in each bucket) and choose u.a.r. a disjoint pairing of the $3n$ points. If no pair contains two points from the same bucket (which represents a loop) and no two pairs contain four points from just two buckets (which represents a multiple edge), this represents a simple cubic graph on n vertices with no loops and no multiple edges. The buckets represent the vertices of the randomly generated cubic graph and each pair represents an edge whose end-points are given by the buckets of the points in the pair. With probability bounded below by a positive constant, loops and multiple edges do not occur (see, for example, Wormald [11, Section 2.2]).

Generating a random cubic graph in this way may be considered as follows. Initially, all vertices have degree 0. Throughout the execution of the generation process, vertices will increase in degree until the generation is complete and all vertices have degree 3. We refer to the graph being generated throughout this process as the *evolving graph*.

3.2 Analysis Using Differential Equations

One method of analysing the performance of a randomised algorithm is to use a system of differential equations to express the expected changes in variables describing the state of the algorithm during its execution. Wormald [12] gives an exposition of this method and in [4] this method is applied to various other graph-theoretic optimisation problems.

In order to analyse our algorithm using a system of differential equations, we incorporate the algorithm as part of a pairing process that generates a random cubic graph. In this way, we generate the random graph in the order that the edges are examined by the algorithm.

During the generation of a random cubic graph we choose the pairs sequentially. The first point, p_i, of a pair may be chosen by any rule. The freedom of choice of p_i enables us to select it u.a.r. from the vertices of given current degree in the evolving graph. In order to ensure that the cubic graph is generated u.a.r., the second point, p_j, of a pair must be selected u.a.r. from all the remaining free (i.e. unpaired) points. We refer to selecting p_j as choosing a *mate* for p_i. Using $B(p_k)$ to denote the bucket that the point p_k belongs to, we say that the edge from $B(p_i)$ to $B(p_j)$ is *exposed* and we say that $B(p_j)$ is *hit* by this exposed edge. Note that we may then determine the current degree of the vertex represented by the bucket $B(p_j)$ without exposing any further edges.

The incorporated algorithm and pairing process may be loosely summarised as follows. Repeatedly select a vertex, u, u.a.r. from those vertices of given current degree in the evolving graph and expose all remaining edges incident with u. This is achieved by selecting each free point u.a.r. from the bucket corresponding to u and selecting a mate for each of these points u.a.r. from all the remaining free points in the evolving graph. The choice of whether to add a vertex to the set under construction will depend on the current degrees of the vertices hit by these exposed edges. Further edges may then be exposed. More detail is given in the following section.

In what follows, we denote the set of vertices of current degree i of the evolving graph, at time t, by $V_i = V_i(t)$ and let $Y_i = Y_i(t)$ denote $|V_i|$. We may express the state of the evolving graph at any point during the execution of the algorithm by considering Y_0, Y_1 and Y_2. In order to analyse our randomised algorithm for finding a 2-independent set, $\mathcal{I}$, of cubic graphs, we calculate the expected change in this state over one unit of time (a unit of time is defined more clearly in Section 5) in relation to the expected change in the size of $\mathcal{I}$. Let $I = I(t)$ denote $|\mathcal{I}|$ at any stage of the algorithm (time t) and let $\mathbf{E}\Delta X$ denote the expected change in a random variable X conditional upon the history of the process. The equations representing $\mathbf{E}\Delta Y_i$ and $\mathbf{E}\Delta I$ are then used to derive a system of differential equations. The solutions to the differential equations describe functions which represent the behaviour of the variables Y_i. Wormald [12, Theorem 6.1] describes a general result which guarantees that the solutions of the differential equations almost surely approximate the variables Y_i. The expected size of the 2-independent set may be deduced from these results.

4 The Algorithm

In Figure 1 we present our algorithm combined with a pairing process. This combination generates an n-vertex cubic graph, G, u.a.r. and, at the same time, finds a subset, $\mathcal{I}$, of the vertices of G.

In the algorithm, the function **isolate**(u) involves the process of exposing all the remaining edges incident with the vertex u and then exposing all the remaining edges incident with the vertices hit by the edges exposed from u. Note that all vertices chosen to be part of the 2-independent set were in V_0 at the start of the operation that selected them. This, and the fact that **isolate**(u) is applied to all such vertices, ensures that the set returned is 2-independent.

```
select $u$ u.a.r. from $V_0$;
$\mathcal{I} \leftarrow u$;
**isolate**($u$);


while ($Y_1 + Y_2 > 0$) do
    if ($Y_2 > 0$)
        select $u$ u.a.r. from $V_2$;
        expose an edge from $u$ to a vertex $v$, say;
        if ($v \in V_1$)
            $\mathcal{I} \leftarrow \mathcal{I} \cup v$;
            isolate($v$);
        endif
    else
        select $u$ u.a.r. from $V_1$;
        expose two edges from $u$ to vertices $w$ and $w'$, say;
        if      ($w \in V_2 \wedge w' \in V_2$)   $v \leftarrow$ NULL;
        else if ($w \in V_1 \wedge w' \in V_2$)   $v \leftarrow w$;
        else if ($w \in V_2 \wedge w' \in V_1$)   $v \leftarrow w'$;
        else                                  select $v$ u.a.r. from $\{w, w'\}$;
        endif
        if ($v \neq$ NULL)
           $\mathcal{I} \leftarrow \mathcal{I} \cup v$;
           isolate($v$);
        endif
    endif
enddo
```

Figure 1: Combined Algorithm and Pairing Process

The algorithm terminates when there are no remaining vertices of degree 1 or 2, which means that a connected component has been completely generated and a 2-independent set has been found in that component. It is well known that cubic graphs are a.a.s. connected, so the result is a.a.s. a 2-independent set in the whole graph.

We select the first element, u, of $\mathcal{I}$ u.a.r. from all of the vertices in the evolving graph, expose all of its incident edges and expose all edges incident with the vertices hit by the edges exposed from u. We say that the remainder of the combined algorithm and pairing process proceeds in *operations* where each operation is denoted by one iteration of the while loop. There are two basic types of operation. A Type 1 operation refers to an operation where $Y_2 = 0$ and a vertex, u, is selected u.a.r. from

V_1. Similarly, a Type 2 operation refers to an operation where $Y_2 > 0$ and a vertex, u, is selected u.a.r. from V_2.

For both types of operation we expose all remaining edges incident with u and investigate the degree(s) of the new neighbour(s) of u. If u has a new neighbour of degree 1, we add this vertex to the 2-independent set (choosing randomly in the case there is more than one). Otherwise, we simply start a new operation. If a vertex, v, is selected by an operation for inclusion in the 2-independent set, all the remaining edges incident with v are exposed and then we expose all the remaining edges incident with the vertices hit by the edges exposed from v.

Figures 2 and 3 show the configurations that may be encountered by performing operations of Type 1 and Type 2 respectively (a.a.s.). The larger circles represent buckets with the points of that bucket represented by smaller circles. Points that were (without a doubt) free (respectively used up) at the start of an operation are coloured black (respectively white). Other points are shaded. Vertices labelled v are chosen for inclusion in the 2-independent set. Vertices of unknown degree labelled μ will have all their incident edges exposed and we refer to these vertices as *rems* (as, if they are of degree i, they are *rem*oved from the set V_i).

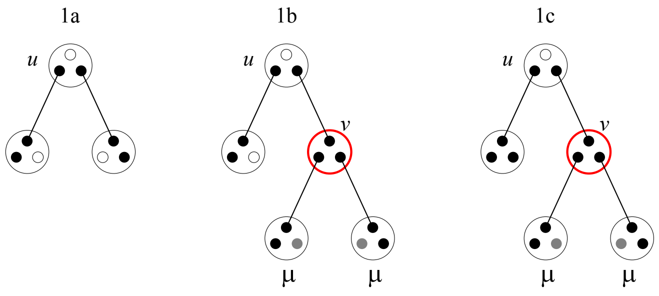

Figure 2: Type 1: Select a vertex from V_1

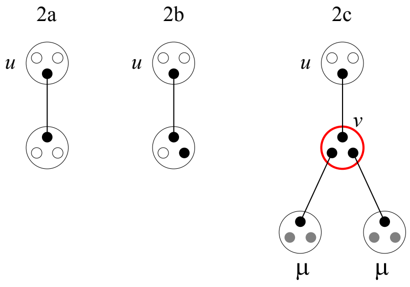

Figure 3: Type 2: Select a vertex from V_2

5 Algorithm Analysis

We analyse the combined algorithm and pairing process using differential equations and in this way prove the following theorem.

Theorem 1 *The size of a maximum* 2*-independent set of a random n-vertex cubic graph is asymptotically almost surely greater than* $0.20485n$.

Proof The first operation of the algorithm involves randomly selecting the first vertex of the 2-independent set and exposing the appropriate edges. We split the remainder of the analysis into two distinct phases. We informally define Phase 1 as the period of time where any vertices in V_2 that are created are used up almost immediately and Y_2 remains small. Once the rate of generating vertices in V_2 becomes larger than the rate that they are used up, the algorithm moves into Phase 2 and all operations involve selecting a vertex from V_2.

We define a *clutch* to be a series of operations in Phase i from an operation of Type i up to but not including the next operation of Type i. We proceed with an examination of each of the two phases, before giving a formal definition of the distinction between the phases. Initially, one only needs to assume that the process begins in Phase 1 and that in Phase 2 there are no operations of Type 1.

5.1 Preliminary Equations For Phase 1

The initial operation of Phase 1 is of Type 1 (at least a.a.s.). A vertex u is chosen u.a.r. from V_1 and all edges incident with u are exposed. Once the degrees of the neighbours of u are known, a vertex may be chosen to be added to the 2-independent set based on the criteria shown by Figure 2. The next operation of the algorithm may be of Type 1 or Type 2 depending on the size of the set V_2. For simplicity, we consider operations of Type 2 first and then combine the equations given by these operations with those given by the operations of Type 1.

Operations of Type 2 in Phase 1 involve the selection of a vertex u from V_2. Let $s = s(t)$ denote the number of free points available in all buckets at a given stage (time t). Note that

$$s = \textstyle\sum_{i=0}^{2}(3-i)Y_i.$$

For our analysis it is convenient to assume that $s > \epsilon n$ for some arbitrarily small but fixed $\epsilon > 0$. Later we discuss the last operations of the algorithm, when $s \leq \epsilon n$.

The expected change in Y_i due to changing the degree of a vertex of degree i from i to $i+1$ by exposing one of its incident edges (at time t) is $\rho_i + o(1)$ where

$$\rho_i = \rho_i(t) = \frac{(i-3)Y_i + (4-i)Y_{i-1}\delta_{i>0}}{s}, \quad 0 \leq i \leq 2.$$

Here, for any statement Q, δ_Q represents 1 if Q evaluates to true and 0 otherwise.

To justify the definition of ρ_i, note that when the point in the vertex of degree i was chosen, the number of points in the buckets corresponding to vertices currently of degree i is $(3-i)Y_i$, and s is the total number of free points. In this case Y_i decreases; it increases if the selected point is from a vertex of degree $i-1$. These two quantities are added because expectation is additive. The term $o(1)$ comes about because the values of all these variables may change by a constant during the course of the operation being examined. Since $s > \epsilon n$ the error is in fact $O(1/n)$.

The expected change in Y_i due to changing the degree of a *rem* from i to 0 and increasing the degrees of its neighbours (other than v) by exposing all remaining edges incident with the *rem* (at time t) is $\mu_i + o(1)$ where

$$\mu_i = \mu_i(t) = \frac{(i-3)Y_i}{s} + \frac{(6Y_0 + 2Y_1)\rho_i}{s}, \quad 0 \le i \le 2.$$

The first term represents the removal of the *rem* from V_i. The expected number of vertices of unknown degree incident with a *rem* is $(6Y_0 + 2Y_1)/s + o(1)$ and each of these will have its degree increased by 1 (giving the second term).

The expected change in Y_i for an operation of Type 2 in Phase 1 (at time t) is $\alpha_i + o(1)$ where

$$\alpha_i = \alpha_i(t) = -\delta_{i=2} + \frac{(i-3)Y_i}{s} + \frac{2Y_1}{s}\delta_{i=2} + \frac{3Y_0}{s}2\mu_i, \quad 0 \le i \le 2.$$

We now consider operations of Type 1. The expected change in Y_i for operation $1h$ given in Figure 2 (at time t) is $\beta_{h,i} + o(1)$ where

$$\beta_{a,i} = \beta_{a,i}(t) = -3\delta_{i=1} + 2\delta_{i=2}, \quad 0 \le i \le 2,$$

$$\beta_{b,i} = \beta_{b,i}(t) = -\delta_{i=0} - 2\delta_{i=1} + \delta_{i=2} + 2\mu_i, \quad 0 \le i \le 2 \quad \text{and}$$

$$\beta_{c,i} = \beta_{c,i}(t) = -2\delta_{i=0} + 2\mu_i, \quad 0 \le i \le 2.$$

For an operation of Type 1 in Phase 1, the neighbours of u (the vertex selected at random from V_1) were in $\{V_0 \cup V_1\}$ before the start of the operation, since $Y_2 = 0$ when the algorithm performs this type of operation. The probability that these neighbours were in V_0 or V_1 are asymptotically $3Y_0/s$ and $2Y_1/s$ respectively. Therefore, the probabilities that, given we are performing an operation of Type 1 in Phase 1, the operation is of type 1a, 1b or 1c are $\mathbf{P}(1a) = 4Y_1^2/s^2 + o(1)$, $\mathbf{P}(1b) = 12Y_0Y_1/s^2 + o(1)$ and $\mathbf{P}(1c) = 9Y_0^2/s^2 + o(1)$ respectively.

We define a *birth* to be the generation of a vertex in V_2 by performing an operation of Type 1 or Type 2 in Phase 1. The expected number of births from a Type 1 operation (at time t) is $\nu_1 + o(1)$ where

$$\nu_1 = \nu_1(t) = 2\mathbf{P}(1a) + (1 + 2\mu_2)\,\mathbf{P}(1b) + 2\mu_2\mathbf{P}(1c).$$

Here, for each case, we consider the probability that vertices of degree 1 become vertices of degree 2 by exposing an edge incident with the vertex. An operation of

Type 1a generates two vertices of degree 2. An operation of Type 1b generates at least one vertex of degree 2 plus possibly more depending on the degree(s) of the other neighbours(s) of the *rems*. The number of births from an operation of Type 1c depends solely on the degree(s) of the other neighbours(s) of the *rems*.

Similarly, the expected number of births from a Type 2 operation (at time t) is $\nu_2 + o(1)$ where

$$\nu_2 = \nu_2(t) = \frac{2Y_1}{s} + 2\mu_2 \frac{3Y_0}{s}.$$

Consider the Type 1 operation at the start of the clutch to be the first generation of a *birth-death* process in which the individuals are the vertices in V_2, each giving birth to a number of children (essentially independent of the others) with expected number ν_2. Then, the expected number in the j^{th} generation is $\nu_1 {\nu_2}^{j-1}$ and the expected total number of births in the clutch is

$$\frac{\nu_1}{1-\nu_2}.$$

For Phase 1, the expected change in Y_i for a clutch is given by

$$\mathbf{E}\Delta Y_i = \mathbf{P}(1a)\beta_{a,i} + \mathbf{P}(1b)\beta_{b,i} + \mathbf{P}(1c)\beta_{c,i} + \frac{\nu_1}{1-\nu_2}\alpha_i + o(1). \tag{1}$$

This assumes $Y_1 + Y_2 \neq 0$, an eventuality which will be discussed later.

The equation giving the expected increase in I for a clutch in Phase 1 is given by

$$\mathbf{E}\Delta I = \mathbf{P}(1b) + \mathbf{P}(1c) + \frac{\nu_1}{1-\nu_2} \times \frac{3Y_0}{s} + o(1) \tag{2}$$

as the contribution to the increase in the size of the 2-independent set by the Type 1 operation in a clutch is 1 if the operation is that of Type 1b or 1c; for each birth we have a Type 2 operation (a.a.s.) and a Type 2 operation increases the size of the 2-independent set if the vertex hit by the edge exposed from u had degree 0 at the start of the operation.

5.2 Preliminary Equations For Phase 2

In Phase 2, all operations are considered to be of Type 2 and therefore a clutch consists of one operation. The expected change in Y_i is given by

$$\mathbf{E}\Delta Y_i = \alpha_i + o(1). \tag{3}$$

and the expected increase in I is given by

$$\mathbf{E}\Delta I = \frac{3Y_0}{s} + o(1). \tag{4}$$

5.3 The Differential Equations

We use the preliminary equations derived in the previous two subsections to formulate a system of differential equations for each phase. Write $Y_i(t) = nz_i(t/n)$, $\mu_i(t) = n\tau_i(t/n)$, $\beta_{j,i}(t) = n\psi_{j,i}(t/n)$, $s(t) = n\xi(t/n)$, $\alpha_i(t) = n\chi_i(t/n)$ and $\nu_j(t) = n\omega_j(t/n)$. From the definitions of μ, β, s, α and ν we have

$$\begin{aligned}
\tau_i &= \tfrac{(i-3)}{\xi} z_i + \tfrac{(6z_0+2z_1)((i-3)z_i+(4-i)z_{i-1}\delta_{i>0})}{\xi^2}, \quad 0 \le i \le 2,\\
\psi_{a,i} &= -3\delta_{i=1} + 2\delta_{i=2}, \quad 0 \le i \le 2,\\
\psi_{b,i} &= -\delta_{i=0} - 2\delta_{i=1} + \delta_{i=2} + 2\tau_i, \quad 0 \le i \le 2,\\
\psi_{c,i} &= -2\delta_{i=0} + 2\tau_i, \quad 0 \le i \le 2,\\
\xi &= \textstyle\sum_{i=0}^{2}(3-i)z_i,\\
\chi_i &= -\delta_{i=2} + \tfrac{(i-3)}{\xi} z_i + \tfrac{2z_1}{\xi}\delta_{i=2} + \tfrac{3z_0}{s} 2\tau_i, \quad 0 \le i \le 2,\\
\omega_1 &= 2\tfrac{4z_1^2}{\xi^2} + (1+2\tau_2)\,\tfrac{12z_0z_1}{\xi^2} + 2\tau_2\tfrac{9z_0^2}{\xi^2} \text{ and}\\
\omega_2 &= \tfrac{2z_1}{\xi} + \tfrac{3z_0}{\xi} 2\tau_2.
\end{aligned}$$

Equation (1) representing $\mathbf{E}\Delta Y_i$ for processing a clutch in Phase 1 forms the basis of a differential equation. The differential equation suggested is

$$\frac{dz_i}{dx} = \frac{4z_1^2}{\xi^2}\psi_{a,i} + \frac{12z_0z_1}{\xi^2}\psi_{b,i} + \frac{9z_0^2}{\xi^2}\psi_{c,i} + \frac{\omega_1}{1-\omega_2}\chi_i, \quad 0 \le i \le 2, \tag{5}$$

where differentiation is with respect to x and xn represents the number, t, of clutches.

Equation (2) representing $\mathbf{E}\Delta I$ after processing a clutch in Phase 1 and writing $I(t) = nz(t/n)$ suggests the differential equation for z as

$$\frac{dz}{dx} = \frac{9z_0^2}{\xi^2} + \frac{12z_0z_1}{\xi^2} + \frac{\omega_1}{1-\omega_2} \times \frac{3z_0}{\xi}. \tag{6}$$

For Phase 2, Equation (3) representing $\mathbf{E}\Delta Y_i$ for processing a clutch suggests the differential equation

$$\frac{dz_i}{dx} = \chi_i, \quad 0 \le i \le 2. \tag{7}$$

Equation (4) representing $\mathbf{E}\Delta I$ after processing a clutch in Phase 2 suggests the differential equation

$$\frac{dz}{dx} = \frac{3z_0}{\xi}. \tag{8}$$

The solution to these systems of differential equations represents the cardinalities of the sets V_i and $\mathcal{I}$ (scaled by $1/n$) for given t. For Phase 1, the equations are (5) and (6) with initial conditions $z_0(0) = 1$, $z_1(0) = 0$, $z_2(0) = 0$ and $z(0) = 0$. The initial conditions for Phase 2 are given by the final conditions for Phase 1 and the equations are given by (7) and (8).

We use a result of [12] to show that during each phase, the functions representing the solutions to the differential equations almost surely approximate the variables Y_i and I with error $o(n)$. For this we need some definitions.

Consider a probability space, S, whose elements are sequences $(q_0, q_1, \ldots)$ where each $q_t \in S$. We use h_t to denote $(q_0, q_1, \ldots, q_t)$, the *history* of the process up to time t, and H_t for its random counterpart. $S^{(n)+}$ denotes the set of all $h_t = (q_0, \ldots, q_t)$ where each $q_i \in S$, $t = 0, 1, \ldots$. All these things are indexed by n and we will consider asymptotics as $n \to \infty$.

We say that a function $f(u_1, \ldots, u_j)$ satisfies a *Lipschitz condition* on $W \subseteq \mathbb{R}^j$ if a constant $L > 0$ exists with the property that

$$|f(u_1, \ldots, u_j) - f(v_1, \ldots, v_j)| \leq L \max_{1 \leq i \leq j} |u_i - v_i|$$

for all $(u_1, \ldots, u_j)$ and $(v_1, \ldots, v_j)$ in W. Note that $\max_{1 \leq i \leq j} |u_i - v_i|$ is the distance between $(u_1, \ldots, u_j)$ and $(v_1, \ldots, v_j)$ in the ℓ^∞ metric.

For variables $Y_1, \ldots, Y_a$ defined on the components of the process, and $W \subseteq \mathbb{R}^{a+1}$, define the *stopping time* $T_W = T_W(Y_1, \ldots, Y_a)$ to be the minimum t such that

$$(t/n, Y_1(t)/n, \ldots, Y_a(t)/n) \notin W.$$

The following is a restatement of [12, Theorem 6.1]. We refer the reader to that paper for explanations, and to Wormald [10] for a similar result with virtually the same proof.

Theorem 2 *Let $\widehat{W} = \widehat{W}(n) \subseteq \mathbb{R}^{a+1}$. For $1 \leq l \leq a$, where a is fixed, let $y_l : S^{(n)+} \to \mathbb{R}$ and $f_l : \mathbb{R}^{a+1} \to \mathbb{R}$, such that for some constant C_0 and all l, $|y_l(h_t)| < C_0 n$ for all $h_t \in S^{(n)+}$ for all n. Let $Y_l(t)$ denote the random counterpart of $y_l(h_t)$. Assume the following three conditions hold, where in (ii) and (iii) W is some bounded connected open set containing the closure of*

$$\{(0, z_1, \ldots, z_a) : \mathbf{P}(Y_l(0) = z_l n, 1 \leq l \leq a) \neq 0 \textit{ for some } n\}\ .$$

(i) For some functions $\beta = \beta(n) \geq 1$ and $\gamma = \gamma(n)$, the probability that

$$\max_{1 \leq l \leq a} |Y_l(t+1) - Y_l(t)| \leq \beta,$$

conditional upon H_t, is at least $1 - \gamma$ for $t < \min\{T_W, T_{\widehat{W}}\}$.

(ii) For some function $\lambda_1 = \lambda_1(n) = o(1)$, for all $l \leq a$

$$|\,\mathbf{E}(Y_l(t+1) - Y_l(t) \mid H_t) - f_l(t/n, Y_1(t)/n, \ldots, Y_a(t)/n)\,| \leq \lambda_1$$

for $t < \min\{T_W, T_{\widehat{W}}\}$.

(iii) Each function f_l is continuous, and satisfies a Lipschitz condition, on

$$W \cap \{(t, z_1, \dots, z_a) : t \geq 0\},$$

with the same Lipschitz constant for each l.

Then the following are true.

(a) For $(0, \hat{z}_1, \dots, \hat{z}_a) \in W$ the system of differential equations

$$\frac{dz_l}{dx} = f_l(x, z_1, \dots, z_a), \qquad l = 1, \dots, a$$

has a unique solution in W for $z_l : \mathbb{R} \to \mathbb{R}$ passing through $z_l(0) = \hat{z}_l$, $1 \leq l \leq a$, and which extends to points arbitrarily close to the boundary of W;

(b) Let $\lambda > \lambda_1 + C_0 n\gamma$ with $\lambda = o(1)$. For a sufficiently large constant C, with probability $1 - O(n\gamma + \frac{\beta}{\lambda}\exp(-\frac{n\lambda^3}{\beta^3}))$,

$$Y_l(t) = nz_l(t/n) + O(\lambda n)$$

uniformly for $0 \leq t \leq \min\{\sigma n, T_{\widehat{W}}\}$ and for each l, where $z_l(x)$ is the solution in (a) with $\hat{z}_l = \frac{1}{n}Y_l(0)$, and $\sigma = \sigma(n)$ is the supremum of those x to which the solution may be extended before reaching within ℓ^∞-distance $C\lambda$ of the boundary of W.

First, we apply Theorem 2 within Phase 1. Define $\widehat{W'}$ to be the vectors for which $z_1 \geq 0$, $z_2 \geq 0$ and $z_1 + z_2 > 0$. Also, for an arbitrarily small $\epsilon > 0$, define W to be the set of all (t, z_0, z_1, z_2, z) for which $t > -\epsilon$, $\xi > \epsilon$, $\omega_2 < 1 - \epsilon$, $z > -\epsilon$ and $z_i < 1 + \epsilon$ where $0 \leq i \leq 2$. Then W defines a domain for the variables t, z_i and z so that Theorem 2 may be applied to the process within Phase 1.

For part (i) of Theorem 2 we must ensure that $Y_i(t)$ does not change too quickly throughout the process. As long as the expected number of births in a clutch is bounded above, the probability of getting say n^ϵ births is $O(n^{-K})$ for any fixed K. This comes from a standard argument as in [12, page 141]. So part (i) of Theorem 2 holds with $\beta = n^\epsilon$ and $\gamma = n^{-K}$. (Note that since $t < T_{\widehat{W'}}$, it follows that $Y_1 + Y_2 > 0$, so that the next operation is of Type 1 or Type 2.) Equations (1) and (2) verify part (ii) of Theorem 2 for a function λ_1 which goes to 0 sufficiently slowly. Note in particular that since $\xi > \epsilon$ inside W, the assumption that $s > \epsilon n$ used in deriving these equations is justified. Part (iii) of Theorem 2 ensures that the rate of change of the variables does not change too quickly in time. By the definition of the phase and the domain W, it may be verified that the functions derived from equations (1) and (2) are continuous on W and its boundary. This implies that the functions are uniformly continuous. From this, the Lipschitz property of the functions required by Theorem 2 part (iii) may be deduced.

The Lipschitz condition in Theorem 2 part (iii) prevents us from choosing a domain which extends to the natural end of the phase which may occur at some time t_2, say. We choose a domain which the variables will almost surely remain inside until time $t_1 = t_2 - \epsilon n$. We may also eliminate a set of undesirable states, which we characterise by $Y_1 + Y_2 \le 0$.

The conclusion of Theorem 2 therefore holds for the process within Phase 1. This implies that with probability $1 - O(n^{1-K} + n^{\epsilon}\lambda^{-1}\exp(-n^{1-3\epsilon}\lambda^3))$, the random variables Y_i and I a.a.s. remain within $O(\lambda n)$ of the corresponding deterministic solutions to the differential equations (5) and (6) until a point arbitrarily close to where it leaves the set W', or until $t = T_{\widehat{W}}$ if that occurs earlier. Note that the latter may only occur when the algorithm has completely processed a component of the graph and a random cubic graph is a.a.s. connected. Choosing $K = 2$ and $\lambda = n^{\epsilon - 1/4}$, say, ensures that with probability $1 - o(n)$, the differential equations for Phase 1 almost surely approximate the variables Y_i and I with error $o(n)$.

We compute the ratio dz_i/dz and we have

$$\frac{dz_i}{dz} = \frac{\frac{4z_1^2}{\xi^2}\psi_{a,i} + \frac{12z_0z_1}{\xi^2}\psi_{b,i} + \frac{9z_0^2}{\xi^2}\psi_{c,i} + \frac{\omega_1}{1-\omega_2}\chi_i}{\frac{9z_0^2}{\xi^2} + \frac{12z_0z_1}{\xi^2} + \frac{\omega_1}{1-\omega_2} \times \frac{3z_0}{\xi}.}, \qquad 0 \le i \le 2,$$

where differentiation is with respect to z and all functions may be taken as functions of z. By solving (numerically) this system of differential equations, we find that the solution hits a boundary of the domain at $\omega_2 = 1 - \epsilon$ (for $\epsilon = 0$ this would approximately be when $z \ge 0.1031$). We now formally define Phase 1 as the period of time from time t=0 to the time t_0 such that $z = t_0/n$ is the solution of ω_2=1.

Our next aim is to show that by the time $\epsilon' n$ operations after the start of Phase 2 (for some $\epsilon' > 0$), the variable Y_2 is a.a.s. at least some constant times n. For this, the main requirement is that the variable ν_2 increases significantly above 1, since $\nu_2 - 1$ is the expected increase in Y_2 when processing a vertex of V_2.

Unfortunately, the expected increase in ν_2 due to processing a vertex from V_1 right near the end of Phase 1 is negative. So instead we consider the variable $\widehat{\nu_2}$ defined by setting $Y_2 = 0$ in the definitions of all variables; that is,

$$\widehat{\nu_2} = \widehat{\nu_2}(t) = \frac{3Y_0}{\hat{s}}2\widehat{\mu_2} + \frac{2Y_1}{\hat{s}}$$

where

$$\begin{aligned}\widehat{\mu_2} &= \widehat{\mu_2}(t) = \frac{(6Y_0 + 2Y_1)\widehat{\rho_2}}{\hat{s}}, \\ \widehat{\rho_2} &= \widehat{\rho_2}(t) = \frac{2Y_1}{\hat{s}} \text{ and} \\ \hat{s} &= 3Y_0 + 2Y_1.\end{aligned}$$

Regarding $\widehat{\nu_2}$ as a function of Y_0 and Y_1 only, we may compute the expected increase in $\widehat{\nu_2}$ due to an operation of Type 1 as

$$\frac{\partial\widehat{\nu_2}}{\partial Y_0}E_0 + \frac{\partial\widehat{\nu_2}}{\partial Y_1}E_1 \tag{9}$$

where E_i is the expected increase in Y_i in such an operation. The latter may be computed from the first three terms on the right hand side of (1). Plugging in the values of Y_0 and Y_1 at the end of Phase 1 gives a positive quantity, approximately 5.14. For a Type 2 operation, the same calculation is used, but the values of E_0 and E_1 come from α_i as seen in (1). The result is 4.55.

Since the formula given by (9) is Lipschitz, it must remain positive for at least $\epsilon_1 n$ operations after reaching time $t_0 - \epsilon n$, for ϵ_1 sufficiently small. Subject to the choice of ϵ_1, we may take ϵ arbitrarily small. It now follows by the usual large deviation argument that the increase in $\widehat{\nu_2}$ between time $t_0 - \epsilon n$ and a time t_1 when $\epsilon_1 n$ operations have occurred in Phase 2 is a.a.s. at least c for some positive constant c. By choosing ϵ sufficiently small, ν_2 is a.a.s. arbitrarily close to 1 at time $t_0 - \epsilon n$, and so the same goes for $\widehat{\nu_2}$ since Y_2 is a.a.s. very small in Phase 1. Thus $\widehat{\nu_2} > 1 + c_1$ a.a.s. at time t_1 for some $c_1 > 0$.

Once this value of $\widehat{\nu_2}$ is attained, since $\widehat{\nu_2} = \nu_2$ when $Y_2 = 0$ we may choose a $c > 0$ such that either $Y_2 > cn$ or $\nu_2 > 1 + c$. In the former case we are well into Phase 2 in the informal sense. In the latter case, due to the Lipschitz property of ν_2, for the next $\epsilon_2 n$ operations, processing a vertex from V_2 produces an expected $1 + c/2$ new vertices of V_2. Again, using the usual large deviation argument, this ensures that with high probability the process moves in the next $\epsilon_2 n$ operations into a state where $V_2 > c_2 n$, and is thus, again, firmly entrenched in Phase 2 in the informal sense. Thus, in either case, there will be some time t_2 which is followed by $c_2 n$ consecutive operations of Type 2, which means that the equations for Phase 2 are valid.

For Phase 2 and for arbitrary small ϵ, define W' to be the set of all (t, z_0, z_1, z_2, z) for which $t > t_2 - \epsilon$, $\xi > \epsilon$, $z > -\epsilon$ and $z_i < 1 + \epsilon$ where $0 \le i \le 2$. Theorem 2 applies as in Phase 1 (with time shifted by subtracting t_2) except that here, a clutch consists of just one operation of Type 2. Note also that the starting point of the process is randomised, which is permitted in Theorem 2.

For part (i) of Theorem 2 we must ensure that $Y_i(t)$ does not change too quickly throughout the process. As a clutch in Phase 1 consists of just one operation, the expected change in any of the variables Y_i for a clutch is at most a constant (in fact Y_0 may decrease by at most 7). So part (i) of Theorem 2 holds with $\beta = 7$ and $\gamma = 0$. Parts (ii) and (iii) of Theorem 2 may be deduced in a similar manner to those for Phase 1.

The conclusion of Theorem 2 therefore holds for the process within Phase 2. This implies (taking $\lambda = o(1)$ tending to 0 sufficiently slowly) that with probability $1 - O(\lambda^{-1}\exp(-n\lambda^3))$, the random variables Y_i and I a.a.s. remain within $O(\lambda n)$ of the corresponding deterministic solutions to the differential equations (7) and (8) until a point arbitrarily close to where it leaves the set W. Choosing, for example, $\lambda = n^{-1/4}$, makes this success probability $1 - o(n)$ and the error $o(n)$.

Computing the ratio dz_i/dz gives

$$\frac{dz_i}{dz} = \frac{\chi_i}{\frac{3z_0}{\xi}}, \quad 0 \le i \le 2.$$

By solving this we see that the solution hits a boundary of W' at $\xi = \epsilon$. Theorem 2 ensures that the random variables Y_i and I in Phase 2 a.a.s. remain within $o(n)$ of the corresponding deterministic solutions to the differential equations (7) and (8) until a point arbitrarily close to where it leaves the domain W'.

From the point in Phase 2 after which Theorem 2 does not apply until the completion of the algorithm, the change in each variable per step is bounded by a constant. Hence, letting ϵ tend to 0 sufficiently slowly, in $o(n)$ steps the change in the random variables Y_i and I is $o(n)$.

The differential equations were solved using a Runge-Kutta method, giving $\xi = \epsilon$ in Phase 2 at $z > 0.20485$. This corresponds to the size of the 2-independent set (scaled by $1/n$) when all vertices are used up, thus proving the theorem. □

6 Remarks

The existence proofs recently shown by Assiyatun [1] that bound the size of a maximum 2-independent set of a random d-regular graph imply that the size of a maximum 2-independent set, I, of a random n-vertex cubic graph a.a.s. satisfies $0.2315n \leq |I| \leq 0.2356n$. However, there is no algorithm that guarantees to find a 2-independent set of a random n-vertex cubic graph that is a.a.s. at least $0.2315n$.

We have presented a simple, yet efficient, polynomial time heuristic for finding a large 2-independent set, $\mathcal{I}$, of cubic graphs and shown that the expected size of $\mathcal{I}$ is a.a.s. greater than $0.20485n$.

References

[1] H. Assiyatun, *Large Subgraphs of Regular Graphs.* Doctoral Thesis, Department of Mathematics and Statistics, The University of Melbourne, Australia, 2001.

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

[3] R. Diestel, *Graph Theory.* Springer-Verlag, 1997.

[4] W. Duckworth, *Greedy Algorithms and Cubic Graphs.* Doctoral Thesis, Department of Mathematics and Statistics, The University of Melbourne, Australia, 2001.

[5] M. Hota, M. Pal and T.K. Pal, An Efficient Algorithm for Finding a Maximum Weight k-Independent Set on Trapezoid Graphs. *Computational Optimization and Applications* **18**(1) (2001), 49–62.

[6] S. Janson, T. Łuczak and A. Rucinski, *Random Graphs.* Wiley, 2000.

[7] M.C. Kong and Y. Zhao, On Computing Maximum k-Independent Sets. *Congressus Numerantium* **95** (1993), 47–60.

[8] M.C. Kong and Y. Zhao, Computing k-Independent Sets for Regular Bipartite Graphs. *Congressus Numerantium* **143** (2000), 65–80.

[9] S. Kutten and D. Peleg, Fast Distributed Construction of Small k-dominating Sets and Applications. *Journal of Algorithms* **28**(1) (1998), 40–66.

[10] N.C. Wormald, Differential Equations for Random Processes and Random Graphs. *Annals of Applied Probability* **5** (1995), 1217–1235.

[11] N.C. Wormald, Models of Random Regular Graphs. In *Surveys in Combinatorics, (Canterbury 1999)*, Cambridge University Press, Cambridge, 1999, 239–298.

[12] N.C. Wormald, The Differential Equation Method for Random Graph Processes and Greedy Algorithms. In *Lectures on Approximation and Randomized Algorithms*, PWN, Warsaw, 1999, 73–155. M. Karoński and H-J. Prömel (editors).

(Received 14/8/2001; revised 23/5/2002)