computation

Article

Large Independent Sets on Random *d*-Regular Graphs with Fixed Degree *d*

Raffaele Marino [1,*] and Scott Kirkpatrick [2]

[1] Dipartimento di Fisica e Astronomia, Università degli Studi di Firenze, Via Giovanni Sansone 1, Sesto Fiorentino, 50019 Firenze, Italy
[2] School of Computer Science and Engineering, The Hebrew University of Jerusalem, Edmond Safra Campus, Givat Ram, Jerusalem 91904, Israel; kirk@cs.huji.ac.il
* Correspondence: raffaele.marino@unifi.it

Abstract: The maximum independent set problem is a classic and fundamental combinatorial challenge, where the objective is to find the largest subset of vertices in a graph such that no two vertices are adjacent. In this paper, we introduce a novel linear prioritized local algorithm tailored to address this problem on random d-regular graphs with a small and fixed degree d. Through exhaustive numerical simulations, we empirically investigated the independence ratio, i.e., the ratio between the cardinality of the independent set found and the order of the graph, which was achieved by our algorithm across random d-regular graphs with degree d ranging from 5 to 100. Remarkably, for every d within this range, our results surpassed the existing lower bounds determined by theoretical methods. Consequently, our findings suggest new conjectured lower bounds for the MIS problem on such graph structures. This finding has been obtained using a prioritized local algorithm. This algorithm is termed 'prioritized' because it strategically assigns priority in vertex selection, thereby iteratively adding them to the independent set.

Keywords: independent set; optimization; lower bounds

Citation: Marino, R.; Kirkpatrick, S. Large Independent Sets on Random *d*-Regular Graphs with Fixed Degree *d*. *Computation* **2023**, *11*, 206. https://doi.org/10.3390/computation11100206

Academic Editors: Franco Bagnoli and Anna T. Lawniczak

Received: 2 September 2023
Revised: 2 October 2023
Accepted: 9 October 2023
Published: 17 October 2023

1. Introduction

Given a graph $G(\tilde{\mathcal{N}}, E)$, where $\tilde{\mathcal{N}}$ is the set of vertices of cardinality $|\tilde{\mathcal{N}}| = N$, and E is the set of edges of cardinality $|E| = M$, finding the maximum set of vertices wherein no two of which are adjacent is a very difficult task. This problem is known as the maximum independent set problem (MIS). The maximum independent set problem can be visualized as a quest to find the largest group of non-neighboring vertices within a graph. Imagine a party where guests represent vertices and the friendships between them represent edges. The MIS is akin to inviting the maximum number of guests such that no two of them are friends, thus ensuring no prior friendships exist within this subset of guests. In graph theoretic terms, it seeks to identify the largest subset of vertices in which no two vertices share an edge. This problem has broad implications and applications, ranging from network design, scheduling, and even in areas such as biology, where one may wish to determine the maximum set of species in a habitat without competition. It was shown to be NP-hard, and no known polynomial algorithm can be guaranteed to solve it [1]. In other words, finding a set $\mathcal{I}$ of vertices, with the maximum cardinality, such that for every two vertices $i, j \in \mathcal{I}$, there is no edge connecting the two, i.e., $(i, j) \notin E$, needs a time that is super-polynomial if P $\neq$ NP.

For example, the first nontrivial exact algorithm for the MIS was due to Tarjan and Trojanowski's $O(2^{N/3}) \sim O(1.2599^N)$ algorithm in 1977 [2]. Since then, many improvements have been obtained. Today, the best algorithm that can solve the MIS exactly needs a time $O(1.1996^N)$ [3]. Those results are bound obtained in the worst-case scenario [4]. We direct the interested reader to [3], and the references therein, for a complete discussion on exact algorithms.

The MIS is important for applications in computer science, operations research, and engineering via such uses as graph coloring, assigning channels to the radio stations, register allocation in a compiler, artificial intelligence etc. [5–8].

In addition to having several direct applications [9], the MIS is closely related to another well-known optimization problem, the maximum clique problem [10,11]. In order to find the maximum clique (the largest complete subgraph) of a graph $G(\tilde{\mathcal{N}}, E)$, it suffices to search for the maximum independent set of the complement of $G(\tilde{\mathcal{N}}, E)$.

The MIS has been studied on many different random structures, in particular on Erdős-Rényi graphs (ER) and random d-regular graphs (RRG). An Erdős-Rényi graph $G_{ER}(N, p)$ is a graph that is selected from the distribution of all graphs of order N, where two different vertices are connected to each other via a probability p. A random d-regular graph is a graph that is selected from the distribution of all d-regular graphs on N vertices, with Nd being even. A d-regular graph is defined as a graph where each vertex has the same number of neighbors, i.e., d.

For the Erdős-Rényi class, with $p = 0.5$, known local search algorithms can find solutions at a rate of only up to half the maximum independent set present, which is $\sim 2\log_{1/(1-p)} N$ [12] in the limit $N \to \infty$.

This behavior also appears for random d-regular graphs $G_d(N)$.

1.1. Related Works

For example, Gamarnik and Sudan [13] showed that, for a sufficiently large value of d, local algorithms cannot find the size of the largest independent set in a d-regular graph of a large girth with an arbitrarily small multiplicative error.

The result of Gamarnik and Sudan [13] was improved by Rahman and Virág [14], who analyzed the intersection densities of many independent sets in random d-regular graphs. They proved that for any $\epsilon > 0$, local algorithms cannot find independent sets in random d-regular graphs with an independence ratio larger than $(1+\epsilon)\frac{\ln d}{d}$ if d is sufficiently large. The independence ratio is defined as the density of the independent set; thus, $\alpha = |\mathcal{I}|/|\tilde{\mathcal{N}}|$. Recently, the exact value of the independence ratio for all sufficiently large d values was given by Ding et al. [15].

However, these results appear to say nothing about small and fixed d values. When d is small and fixed, e.g., $d = 3$ or $d = 30$, indeed, only lower and upper limits, expressed in terms of the independence ratio, are known.

Lower bounds on the independent sets' size identify sets that an efficient algorithm can find, while upper bounds are on the actual maximum independent set, not just on the size an algorithm can find.

The first upper bound for such a problem was given in 1981 by Bollobás [16]. He showed that the supremum of the independence ratio of 3-regular graphs with large girths was less than $6/13 \sim 0.461538$ in the limit of $N \to \infty$.

McKay, in 1987, improved and generalized this result to d-regular graphs with large girths, for different values of d [17], by using the same technique and a much more careful calculation. For example, for the cubic graph (the 3-regular graph), he was able to push Bollobás upper bound down to 0.455370. However, since then, only for cubic graphs, the upper bound has been improved by Balogh et al. [18], namely, to 0.454. Cavity methods suggest a slightly lower upper bound and, thus, a smaller gap at small values of d [19]. For example, the upper bound given in [19] for $d = 3$ was 0.4509, while for $d = 4$, it was 0.4112. In [15], it was shown that this approach can be rigorously proven, but again, only for large d values. Recently, however, this approach has been proven for $d \leq 19$ in [20].

Remarkable results for lower bounds were first obtained by Wormald in 1995 [21]. He considered processes in which random graphs are labeled as they are generated and derived the conditions under which the parameters of the process concentrate around the values that come from the solution of an associated system of differential equations, which are equations for the populations of various configurations in the graph as it is grown. By solving the differential equations, he computed the lower bounds for any fixed

d returned by a prioritized algorithm, thereby improving the values of the bounds given by Shearer [22].

This algorithm is called prioritized, because there is a priority in choosing vertices added to the independent set [23]. It follows the procedure of choosing vertices in the independent set $\mathcal{I}$ one by one, with the condition that the next vertex is chosen randomly from those with the maximum number of neighbors adjacent to the vertices already in $\mathcal{I}$. After each new vertex in $\mathcal{I}$ is chosen (or labeled with an I), we must complete all of its remaining connections and label the neighbors, which are identified as members of the set $\mathcal{V}$ (for vertex cover). Although each vertex in $\mathcal{I}$ can be chosen according to its priority, the covering vertices that complete its unfilled connections must then be chosen at random among the remaining connections to satisfy Bolobas' configuration model [21].

This priority is a simple way to minimize the size of the set of covered vertices and maximize the number of sites remaining as candidates for the set $\mathcal{I}$. More precisely, we are given a random d-regular graph $G_d(N)$, and we randomly choose a site i from the set of vertices $\tilde{\mathcal{N}}$. We set i into $\mathcal{I}$, and we set all of the vertices neighboring i into a set $\mathcal{V}$. We label the elements of $\mathcal{I}$ with the letter I, while the elements of $\mathcal{V}$ are labeled with the letter V. Then, from the subset of vertices in $\tilde{\mathcal{N}}$ that are neighbors of vertices in $\mathcal{V}$, but are not yet labeled I or V, we randomly choose the element k that has the maximum number of connections with sites in $\mathcal{V}$. We set it into $\mathcal{I}$. The vertices neighboring k, which are not in $\mathcal{V}$, are added to the set $\mathcal{V}$. This rule is repeated until $|\mathcal{I}| + |\mathcal{V}| = N$. Along with this algorithm, one can consider an associated algorithm that simultaneously generates the random d-regular graph $G_d(N)$ and labels vertices with the letter I or V. This associated algorithm, which will be described in detail in the next sections, allowed Wormald to build up the system of differential equations used for computing lower bounds for the MIS.

Improvements on this algorithm were achieved by Duckworth et al. [24]. These improvements were obtained by observing, broadly speaking, that the size of the structure produced by the algorithm is almost the same for d-regular graphs of very large girths as it is for a random d-regular graph. However, since then, new lower bounds have been achieved only at small values of d, e.g., $d = 3$ and $d = 4$. Interesting results at $d = 3$ have been achieved by Csóka, Gerencsér, Harangi, and Virág [25]. They were able to find an independent set of cardinality of up to $0.436194\,N$ using invariant Gaussian processes on the infinite d-regular tree. This result was once again improved by Csóka [26] alone, who was able to increase the cardinality of the independent set on large-girth 3-regular graph by up to $0.445327N$ and on a large-girth 4-regular graph by up to $0.404070N$, by numerically solving the associated system of differential equations.

These improvements were obtained by deferring the decision as to whether a site $i \in \tilde{\mathcal{N}}$ must be labeled with the letter I or V. More precisely, this requires that the sites for which a decision is deferred need additional (temporary) labels. This means that counting the evolution of their populations, either through a differential equation or with an experiment, becomes more complicated.

Csóka [26] was able to improve the lower bounds for $d = 3$ and $d = 4$, but his method was not applicable to values of $d > 4$. These bounds cannot be considered fully rigorous, as they require some kind of computer simulation or estimation [20].

1.2. Main Results

This paper aims to compute independent set density for any $d \geq 5$ using an experimental approach, i.e., algorithms that are linear in N. Table 1 presents the best upper and lower bounds for $d \in [5, 100]$ in the first and second columns, respectively. Recently, the authors in [27] presented a Monte Carlo method that can experimentally outperform any algorithm in finding a large independent set in random d-regular graphs, in a (using the words of the authors) "*running time growing more than linearly in N*" [27]. These authors conjectured lower bound improvements only for $d = 20$ and $d = 100$, but with experimental results obtained on random d-regular graphs of the order $N = 5 \cdot 10^4$. However, in this work, we are interested in comparing our results with the ones given by the family of prioritized algorithms, because we believe that a rigorous analysis of the computational complexity should be performed on these types of algorithms.

In this paper, as stated above, we present the experimental results of a greedy algorithm, i.e., a deferred decision algorithm built upon existing heuristic strategies, which leads to improvements on the known lower bounds of a large independent set in random d-regular graphs $\forall d \in [5, 100]$ [21,24,28]. These bounds cannot be considered fully rigorous, as they require computer simulations and estimations. However, the main contribution of this manuscript is to present a new method that is able to return, on average, independent sets for random d-regular graphs that before were not possible. This new algorithm runs in linear time $O(N)$ and melds Wormald's, Duckworth's and Zito's, and Csoka's ideas of prioritized algorithms [21,24,26,28]. The results obtained here are conjectured new lower bounds for a large independent set in random d-regular graphs. They were obtained by inferring the asymptotic values that our algorithm can reach when $N \to \infty$ and by averaging sufficient simulations to achieve confidence intervals at 99%. These results led to improvements on the known lower bounds $\forall d \in [5, 100]$ that, as far as we know, have not been reached by any other greedy algorithm (see fourth column Table 1). Although the gap regarding upper bounds is still present, these improvements may imply new rigorous results for finding a large independent set in random d-regular graphs.

Table 1. The table shows the values of upper and lower bounds for the independence ratio for random d-regular graphs with small and fixed values of d. d is the degree of the random d-regular graph. The α_{UB} column describes the upper bound computed by Harangi [20] for $5 \leq d \leq 10$ and by Barbier et al. in [19] for $20 \leq d \leq 100$. Upper bounds identify the actual maximum values of the independent set fraction. The α_{LB} column identifies the best independent set density values obtained in [21,24,26,28]. Lower bounds identify the size of independent sets that an efficient algorithm can find. The last column, i.e., $\alpha_\infty \pm z_{99\%}\sigma_{\alpha_\infty}$, instead identifies the 99% confidence intervals of the conjectured new bounds.

d	α_{UB}	α_{LB}	$\alpha_\infty \pm z_{99\%}\sigma_{\alpha_\infty}$
5	0.37917	0.35930	0.36476(2)
6	0.35289	0.33296	0.33600(2)
7	0.33081	0.31068	0.31241(2)
8	0.31192	0.28800	0.29255(1)
9	0.29552	0.27160	0.27555(2)
10	0.28110	0.25730	0.26079(2)
20	0.19480	0.17380	0.17550(4)
50	0.10980	0.09510	0.09574(2)
100	0.06740	0.05720	0.05754(1)

1.3. Paper Structure

The paper is structured as follows: in Section 2, we define our deferred decision algorithm, and we introduce a site labeling, which will identify those sites for which we defer the I/V labeling decisions. In Section 3, we present the deferred decision algorithm for $d = 3$, and we introduce the experimental results obtained on random 3-regular graphs of sizes up to 10^9 as a sanity check of our experimental results. We recall that the order of a graph $G(\tilde{\mathcal{N}}, E)$ is the cardinality of its vertex set $\tilde{\mathcal{N}}$, while the size of a graph $G(\tilde{\mathcal{N}}, E)$ is the cardinality of its edge set E. In Section 4, we present our deferred decision algorithm for $d \geq 5$, and the experimental results associated with it, using extrapolation on random d-regular graphs with sizes of up to 10^9.

2. Notation and the General Operations of the Deferred Decision Algorithm

In this section, we define the notation used throughout this manuscript, and we define all of the operations that will be used to understand the deferred decision algorithm. As a starting point, and also for the following section, we define the set $\mathcal{N} = \tilde{\mathcal{N}}$ as the set of unlabeled nodes. We start by recalling that we deal with random d-regular graphs $G_d(N)$, where d is the degree of each vertex $i \in \mathcal{N}$, where $\mathcal{N}$ is the set of vertices, and $N = |\mathcal{N}|$. All vertices $i \in \mathcal{N}$ are unlabeled.

In order to build a random d-regular graph, we used the method described in [21] and introduced in [16].

Definition 1 (Generator of Random d-Regular Graphs Algorithm). *We take dN points, with dN being even, and distribute them in N urns that are labeled $1, 2, \ldots, N$, with d points in each urn. We choose a random pairing $P = p_1, \ldots, p_{dN/2}$ of the points such that $|p_i| = 2 \forall i$. Each urn identifies a site in $\mathcal{N}$. Each point is in only one pair p_i, and no pair contains two points in the same urn. No two pairs contain four points from just two urns. In order to build a d-regular graph $G_d(N)$, we then connect two distinct vertices i and j if some pair has a point in urn i and one in urn j. The conditions on the pairing prevent the formation of loops and multiple edges.*

The referred pairing must be chosen uniformly at random and subjected to the constraints given. This can be done by repeatedly choosing an unpaired point and then choosing a partner for this point to create a new pair. As long as the partner is chosen uniformly at random from the remaining unpaired points, and as long as the process is restarted if a loop or multiple edge is/are created, the result is a random pairing of the required type [21].

In this paper, we use the method described above so that while we generate the random d-regular graph $G_d(N)$, concurrently with our labeling process, we identify new links as labeling sites.

The graphs built using the **Generator of Random d-Regular Graphs Algorithm** prevent the formation of loops and multiple edges without introducing bias in the distribution where we sample the graphs [21,23].

We define two separate sets $\mathcal{I}$ and $\mathcal{V}$ for independent and vertex cover sites, respectively. $\mathcal{I}$ identifies the set of graph nodes that satisfies the property wherein no two nodes are adjacent, and $\mathcal{V}$ is its complement. A site $i \in \mathcal{I}$ is labeled with the letter I, while a site $j \in \mathcal{V}$ is labeled with the letter V.

We define Δ_i to be the degree of a vertex i, i.e., the number of links that a site is connected with, while $\overline{\Delta}_i$ is the *anti*degree of a vertex i, i.e., the number of free connections that i needs to complete during the graph-building process. Of course, the constraint $\Delta_i + \overline{\Delta}_i = d$ is always preserved $\forall i \in \mathcal{N}$. At the beginning of the graph-building process, all $i \in \mathcal{N}$ have $\Delta_i = 0$, $\overline{\Delta}_i = d$. At the end of the graph-building process, all graph nodes will have $\Delta_i = d$, $\overline{\Delta}_i = 0$. We define ∂i to be the set that contains all of the neighbors of i.

For the sake of clarity, we define a simple subroutine on a single site $i \in \mathcal{N}$ of the **Generator of Random d-Regular Graphs Algorithm** (Subroutine GA(i, $\overline{\Delta}_i$)) that will be useful for describing the algorithm presented in the next sections. The Subroutine GA(i, $\overline{\Delta}_i$) generates the remaining $\overline{\Delta}_i$ connections of site i. It keeps the supporting data to reflect the evolution of the network growth.

Algorithm 1: Subroutine GA(i, $\overline{\Delta}_i$)

```
Input: $i \in \mathcal{N}$, $\overline{\Delta}_i$;
Output: $i$ connected with $d$ sites;
1 Using the rules in Definition 1, $i$ is connected randomly with $\overline{\Delta}_i$-sites;
2 $\overline{\Delta}_i = 0$;
3 $\Delta_i = d$;
4 return $i$ connected with $d$ sites;
```

The sites that we choose following some priority, either the one we describe or any other scheme, will be called P sites. The sites that are found by following links from the P sites (or by randomly generating connections from the P sites) are called C sites. More precisely, each site $j \in \mathcal{N}$ that we choose that is not labeled yet with any letter (C or P), s.t. $\overline{\Delta}_j \leq 2$ and the random connection(s) present on j is(are) on site(s) in $\mathcal{V}$, is a P site.

The set $\mathcal{P}$ defines the set of P sites. The set $\mathcal{P}$ is kept in ascending order with respect to the *anti*degree $\overline{\Delta}_i$ of each site $i \in \mathcal{P}$.

In general, a site i that is labeled as P will be surrounded by two sites that are labeled as C. Because the labeling of those sites is deferred, we call those $C - P - C$ structures *virtual* sites. A single virtual site, $\tilde{v}$, has an antidegree $\overline{\Delta}_{\tilde{v}}$ equal to the sum of all of the antidegrees of sites l that compose site $\tilde{v}$, i.e., $\overline{\Delta}_{\tilde{v}} = \sum_{l \in \tilde{v}} \overline{\Delta}_l$. The number of sites $l \in \tilde{v}$ is equal to the cardinality of $|\tilde{v}|$. The degree of $\tilde{v}$ is $\Delta_{\tilde{v}} = d\,|\tilde{v}| - \overline{\Delta}_{\tilde{v}}$. As an example, we show in Figure 1 the operation of how a virtual site is created from a site $s \in \mathcal{P}$ with $\overline{\Delta}_s = 2$ and $\Delta_s = 1$, as well as two sites $j, k \in \mathcal{N}$ with $\overline{\Delta}_j = 3$, $\overline{\Delta}_k = 3$, $\Delta_j = 0$, and $\Delta_k = 0$. Let us assume that a site $s \in \mathcal{N}$ s.t. $\overline{\Delta}_s = 2$ exists. This is possible because a site $l \in \mathcal{V}$ is connected with it. This means that s must be labeled as P and put into $\mathcal{P}$. Let us run Subroutine GA(s, $\overline{\Delta}_s$) on s and assume that the s connects with two neighbors $j, k \in \mathcal{N}$. Given that $j, k \in \mathcal{N}$ are connected to a P site, they are labeled as C values. We then define $\tilde{v} = \{s, j, k\}$. This set is a virtual node $\tilde{v}$ with $\overline{\Delta}_{\tilde{v}} = 4$.

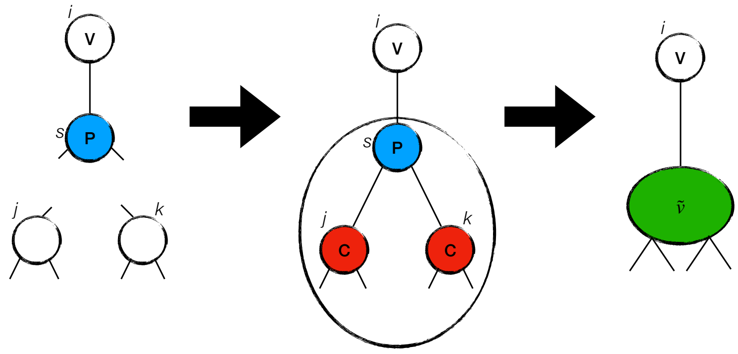

Figure 1. The figure shows the equivalence of a $C - P - C$ structure with a $\tilde{v}$ site of antidegree $\overline{\Delta}_{\tilde{v}} = 4$. The site s, labeled P, is connected to site $i \in \mathcal{V}$ with two random sites, j and k, labeled C values. The resulting structure is a virtual site $\tilde{v} \in \mathcal{A}$.

We define $\mathcal{A}$ to be the set of virtual sites. The set $\mathcal{A}$ is kept in ascending order with respect to the *anti*degree $\overline{\Delta}$ of each virtual site $\tilde{v} \in \mathcal{A}$. Virtual sites can be created—as described above—expanded, or merged together (creating a new virtual site $\tilde{\theta} = \cup_i \tilde{v}_i$). Two examples are shown in Figures 2 and 3.

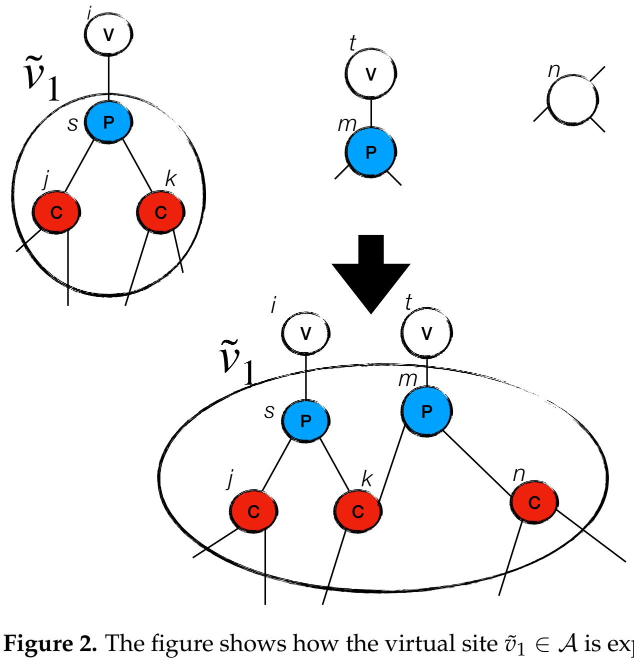

Figure 2. The figure shows how the virtual site $\tilde{v}_1 \in \mathcal{A}$ is expanded.

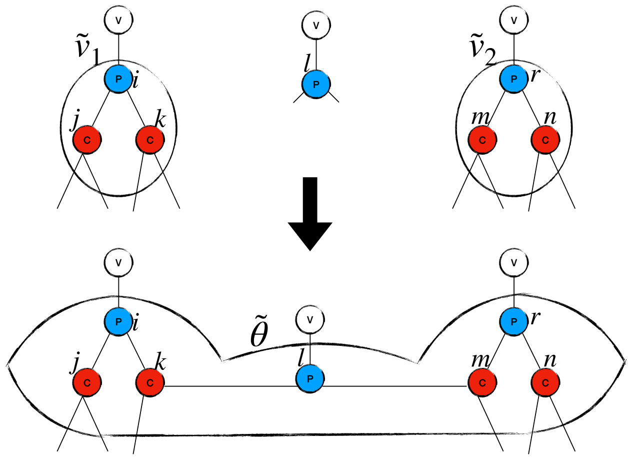

Figure 3. The figure shows how a virtual site $\tilde{\theta} \in \mathcal{A}$ with $\overline{\Delta}_{\tilde{\theta}} = 6$ is created.

Figure 2 shows how to expand a virtual site $\tilde{v}_1$. Let us imagine that a site $m \in \mathcal{P}$, with antidegree $\overline{\Delta}_m = 2$, is chosen. Let us run Subroutine GA$(m, \overline{\Delta}_m)$ on m. Assume that m connects with $n \in \mathcal{N}$ ($\overline{\Delta}_n = 3$) and with $\tilde{v}_1 \in \mathcal{A}$ ($\overline{\Delta}_{\tilde{v}_1} = 4$). In this case, $\tilde{v}_1 \in \mathcal{A}$ expands itself, thereby swallowing sites m and n and having a $\overline{\Delta}_{\tilde{v}_1} = 5$.

Figure 3 shows how two virtual sites merge together. Let us imagine that a site $p \in \mathcal{P}$, with antidegree $\overline{\Delta}_p = 2$, is chosen during the graph-building process. Let us run Subroutine GA$(p, \overline{\Delta}_p)$ on m. Assume that p connects with two virtual sites $\tilde{v}_1 \in \mathcal{A}$ and $\tilde{v}_2 \in \mathcal{A}$, with $\overline{\Delta}_{\tilde{v}_1} = 4$ and $\overline{\Delta}_{\tilde{v}_2} = 4$. The new structure is a virtual site $\tilde{\theta} \in \mathcal{A}$ with $\overline{\Delta}_{\tilde{\theta}} = 6$.

We define in the following a list of operations that will be useful for understanding the algorithm presented in the next sections.

Definition 2 ($OP^i_{move}(i, \mathcal{X}, \mathcal{Y})$)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be two sets. Let $i \in \mathcal{X}$ and $i \notin \mathcal{Y}$. We define $OP^i_{move}(i, \mathcal{X}, \mathcal{Y})$ to be the operation that moves the site $i \in \mathcal{X}$ from the set $\mathcal{X}$ to $\mathcal{Y}$, i.e., $i \in \mathcal{Y}$ and $i \notin \mathcal{X}$.*

For example, $OP^i_{move}(i, \mathcal{N}, \mathcal{V})$ moves $i \in \mathcal{N}$ from the set $\mathcal{N}$ to $\mathcal{V}$, i.e., $i \in \mathcal{V}$ and $i \notin \mathcal{N}$. Instead, the operation $OP^i_{move}(i, \mathcal{N}, \mathcal{I})$ moves $i \in \mathcal{N}$ from the set $\mathcal{N}$ to $\mathcal{I}$, i.e., $i \in \mathcal{I}$ and $i \notin \mathcal{N}$. We recall that when a site is set into $\mathcal{I}$, it is labeled with I, while when a site is set into $\mathcal{V}$, it is labeled with V.

Definition 3 ($OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$)**.** *Let $\mathcal{A}$ be the set that contains the virtual nodes $\tilde{v}$. We define $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$ to be the operation that deletes the site $\tilde{v} \in \mathcal{A}$ from the set $\mathcal{A}$, i.e., the element $\tilde{v} \notin \mathcal{A}$ anymore, and it applies the operation $OP^i_{move}(i, \mathcal{X}, \mathcal{Y})$ on each site $i \in \tilde{v}$ through the following rule:*

- *if $i \in \tilde{v}$ is labeled with the letter P, then $\mathcal{X} = \mathcal{N}$ and $\mathcal{Y} = \mathcal{I}$;*
- *if $i \in \tilde{v}$ is labeled with the letter C, then $\mathcal{X} = \mathcal{N}$ and $\mathcal{Y} = \mathcal{V}$.*

Definition 4 ($SWAP - OP(\tilde{v})$)**.** *Let $\tilde{v} \in \mathcal{A}$. We define $SWAP - OP(\tilde{v})$ to be the operation such that $\forall i \in \tilde{v}$:*

- *if i is labeled as P, then the label P swaps to C;*
- *if i is labeled as C, then the label C swaps to P.*

Figure 4 shows how $SWAP - OP(\tilde{v})$ acts on a virtual site $\tilde{v}$.

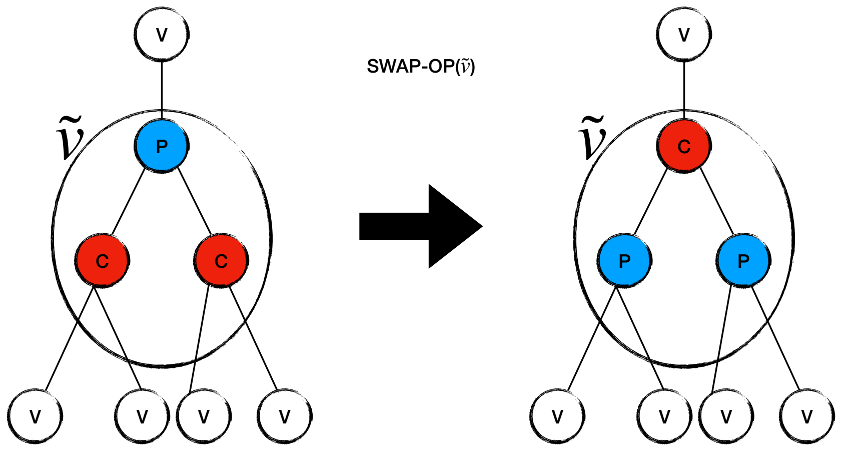

Figure 4. The figure shows how $SWAP - OP(\tilde{v})$ works on a virtual site $\tilde{v}$, which has $\overline{\Delta}_{\tilde{v}} = 0$. $SWAP - OP(\tilde{v})$ swaps all P sites in $\tilde{v}$ into C sites and all C sites in $\tilde{v}$ into P sites.

3. The Deferred Decision Algorithm for $d = 3$

In this section, we present our algorithm, which is simpler and slightly different from the one in [26], but it is based on the same idea for determining a large independent set in random d-regular graphs with $d = 3$, i.e., $G_3(N)$. The algorithm in [26] works only for $d = 3$ and $d = 4$. This exercise is performed as a sanity check for the algorithm. This algorithm will also be at the core of the algorithm developed in Section 4.

As mentioned above, the algorithm discussed in this paper is basically a prioritized algorithm, i.e., an algorithm that makes local choices in which there is a priority in selecting a certain site. Our algorithm belongs to this class.

We start the discussion on the local algorithm for $d = 3$ by giving the pseudocode of the algorithm in Algorithm 2.

The algorithm starts by building up a set of sites $\mathcal{N}$ of cardinality $|\mathcal{N}| = N$, and from the set it randomly picks a site i. Then, the algorithm completes the connections of site i in a random way by following the method described in Algorithm 1. Once all its connections are completed, site i has $\Delta_i = 3$ and $\overline{\Delta}_i = 0$. It is labeled with the letter V, erased from $\mathcal{N}$, and set into $\mathcal{V}$. In other words, operation $OP^{i}_{move}(i, \mathcal{N}, \mathcal{V})$ is applied on it. Each neighbor of i, i.e., $j \in \partial i$, has degree $\Delta_j = 1$ and antidegree $\overline{\Delta}_j = 2$. Therefore, they are set into $\mathcal{P}$ and thus labeled P values.

The algorithm picks a site k from $\mathcal{P}$ with the minimum remaining connections. In general, if k has $\overline{\Delta}_k \neq 0$, the algorithm completes all its connections, and it removes it from $\mathcal{P}$. Each site connected with a P value is automatically labeled with the letter C. If a site $k \in \mathcal{P}$ connects to another site $j \in \mathcal{P}$, with $j \neq k$, j is removed from $\mathcal{P}$, and it is labeled as a C value.

If $k \in \mathcal{P}$ has $\overline{\Delta}_k = 0$, the site k is set into $\mathcal{I}$, and it is removed from $\mathcal{N}$ and $\mathcal{P}$, i.e., the algorithm applies the operation $OP^{k}_{move}(k, \mathcal{N}, \mathcal{I})$.

As defined in Section 2, a $C - P - C$ structure is equivalent to a single virtual site, $\tilde{v}$, which has an antidegree $\overline{\Delta}_{\tilde{v}}$. Each virtual site $\tilde{v}$ is created with $\overline{\Delta}_{\tilde{v}} > 2$, and it is inserted into the set $\mathcal{A}$.

Once the set $\mathcal{P}$ is empty, and if and only if in $\mathcal{A}$ the virtual sites with antidegrees less than or equal to 2 are not present, the algorithm selects a site $\tilde{v} \in \mathcal{A}$ with the largest antidegree $\overline{\Delta}_{\tilde{v}}$, and it applies the operation $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$ after having completed all the connections $\forall i \in \tilde{v}$ with $\overline{\Delta}_i \neq 0$ by using Algorithm 1 on each $i \in \tilde{v}$ with $\overline{\Delta}_i \neq 0$.

We apply operation $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$ on virtual sites $\tilde{v} \in \mathcal{A}$ with the largest antidegree, because we hope the random connections outgoing from those sites will reduce the antidegrees of the existing virtual sites in $\mathcal{A}$ in such a way that the probability of having virtual nodes with antidegrees of $\overline{\Delta}_{\tilde{v}} \leq 2$ increases. In other words, we want to create islands of virtual sites that are surrounded by a sea of V sites in order to apply the $SWAP - OP(\tilde{v})$

on those nodes. This protocol, indeed, makes it possible to increase the independent set cardinality and decrease the vertex cover set cardinality.

Algorithm 2: local algorithm for $d = 3$

```
   Input: $N, d = 3$;
   Output: $\mathcal{I}$;
1  Build the set of sites $\mathcal{N}$ with $|\mathcal{N}| = N$;
2  $\mathcal{I} = \emptyset$;
3  $\mathcal{V} = \emptyset$;
4  Pick a random site $i \in \mathcal{N}$;
5  Run Subroutine $GA(i, \overline{\Delta}_i)$;
6  Apply $OP^{i}_{move}(i, \mathcal{N}, \mathcal{V})$;
7  while $\mathcal{N} \neq \emptyset$ do
8      while $\exists i \in \mathcal{N}$ s.t. $\overline{\Delta}_i \leq 2 \wedge i \notin \mathcal{P} \wedge i$ is not labeled $C$ do
9          Label $i$ with letter $P$ and insert it into $\mathcal{P}$;
10     if $\mathcal{P} \neq \emptyset$ then
11         while $\mathcal{P} \neq \emptyset$ do
12             Pick the first $l \in \mathcal{P}$ (we recall that elements in $\mathcal{P}$ are in ascending order);
13             if $\overline{\Delta}_l = 0$ then
14                 Apply $OP^{l}_{move}(l, \mathcal{N}, \mathcal{I})$;
15                 Remove $l$ from $\mathcal{P}$;
16             else
17                 Run Subroutine $GA(l, \overline{\Delta}_l)$;
18                 If a neighbour $j$ of $l$, i.e., $j \in \partial l$, is in $\mathcal{P}$ remove $j$ from $\mathcal{P}$;
19                 $\forall j \in \partial l$, label each $j$ with the letter $C$;
20                 Build or update the virtual node $\tilde{v}$ and, if it is not present, insert it
                     into $\mathcal{A}$;
21                 Remove $l$ from $\mathcal{P}$;
22     else
23         while $\exists \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 0$ do
24             Apply $SWAP - OP(\tilde{v})$;
25             Apply $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$;
26         while $\exists \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 1$ do
27             Apply $SWAP - OP(\tilde{v})$;
28             For $i \in \tilde{v}$ labeled $P$ s.t. $\overline{\Delta}_i = 1$ run Subroutine $GA(i, \overline{\Delta}_i)$;
29             Pick $j \in \partial i$,with $j$ the last neighbour of $i$ added;
30             Run Subroutine $GA(j, \overline{\Delta}_j)$;
31             Apply $OP^{j}_{move}(j, \mathcal{N}, \mathcal{V})$;
32             Apply $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$;
33         while $\exists \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 2$ do
34             Apply $SWAP - OP(\tilde{v})$;
35             $\forall i \in \tilde{v}$ labeled $P$ s.t. $\overline{\Delta}_i \leq 2$ run Subroutine $GA(i, \overline{\Delta}_i)$ and label the
                 neighbour(s) of $i$ with the letter $C$;
36             Update the virtual node $\tilde{v}$;
37         Pick $\tilde{v}$ s.t. $\max_{\tilde{v} \in \mathcal{A}} \overline{\Delta}_{\tilde{v}}$;
38         $\forall i \in \tilde{v}$ s.t. $\overline{\Delta}_i \neq 0$ and labeled $C$, run Subroutine $GA(i, \overline{\Delta}_i)$;
39         Apply $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$;
40 return $\mathcal{I}$;
```

For this reason, if virtual nodes with antidegrees of $\overline{\Delta}_{\tilde{v}} \leq 2$ exist in $\mathcal{A}$, those sites have the highest priority in being selected. More precisely, the algorithm follows the priority rule:

1. $\forall \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 0$, the algorithm sequentially applies the operation $SWAP - OP(\tilde{v})$ and then the operation $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$.
2. If no virtual sites $\tilde{v} \in \mathcal{A}$ with $\overline{\Delta}_{\tilde{v}} = 0$ are present, then the algorithm looks for those that have $\overline{\Delta}_{\tilde{v}} = 1$. $\forall \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 1$, it applies the operation $SWAP - OP(\tilde{v})$, completes the last connection of the site $i \in \tilde{v}$ with $\overline{\Delta}_i = 1$, applies on the last neighbour of i added to $OP^{j}_{move}(j, \mathcal{N}, \mathcal{V})$, and then applies $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$.
3. If no virtual sites $\tilde{v} \in \mathcal{A}$ with $\overline{\Delta}_{\tilde{v}} = 0$ and $\overline{\Delta}_{\tilde{v}} = 1$ are present, then the algorithm looks for those that have $\overline{\Delta}_{\tilde{v}} = 2$. $\forall \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 2$, it applies the operation $SWAP - OP(\tilde{v})$, completes the last connections of the sites $i \in \tilde{v}$ with $\overline{\Delta}_i \neq 0$, labels the new added sites with the letter C, and updates the degree and the antidegree of the virtual node $\tilde{v}$.

The algorithm proceeds by selecting virtual nodes and creating sites labeled as P values until $\mathcal{N} = \emptyset$. Once $\mathcal{N} = \emptyset$, it returns the set $\mathcal{I}$, which is the set of independent sites. The code of the algorithm can be released upon request.

We are comparing numerical results for independence ratios that agree with theoretical ones, at least up to the fifth digit. For this reason, we performed an accurate analysis on random 3-regular graphs, starting from those that had an order of 10^6 and pushing the number up to $5 \cdot 10^8$.

This analysis aimed to compute the sample mean of the independence ratio size $\alpha(N)$ that was outputted by our algorithm. Each average was obtained in the following manner: for graphs of the order $N = 10^6$, we averaged over a sample of 10^4 graphs; for the order $N = 2.5 \cdot 10^6$, we made an average over a sample of $7.5 \cdot 10^3$ graphs; for the order $N = 5 \cdot 10^6$, we made an average over a sample of $5 \cdot 10^3$ graphs; for the order $N = 10^7$, the average was performed over a sample of 10^3 graphs; we performed this average for the $N = 2.5 \cdot 10^7$ over $7.5 \cdot 10^2$ graphs, for the $N = 5 \cdot 10^7$ over $5 \cdot 10^2$ graphs, for the $N = 10^8$ over 10^2 graphs, for the $N = 2.5 \cdot 10^8$ over 50 graphs, and for the $N = 5 \cdot 10^8$ over 10 graphs. The mean and the standard deviation for each analyzed sample are reported in Table 2. Upon observing that the values of each independent set ratio sample mean reached an asymptotic value, we performed a linear regression on the model $f(N) = (a/\ln N) + \alpha_\infty$ in order to estimate the parameter α_∞ (blue line in Figure 5). When $N \to \infty$, the first term of the regression, i.e., $(a/\ln N)$, went to 0, thereby leaving out the value of α_∞ that describes the asymptotic value of the independence ratio that our algorithm can reach. After adopting the model $f(N) = (a/\ln N) + \alpha_\infty$, one might naturally question the rationale behind the specific choice of $\frac{1}{\ln N}$ as a regressor. This decision is grounded in the intrinsic nature of large graphs. In numerous instances, as seen in small-world networks, the properties of these networks scale logarithmically with their size. The logarithmic factor effectively translates the expansive range of data values into a scale that is more analytically tractable. By employing $\frac{1}{\ln N}$ as our regressor, we are capturing the progressively diminishing influence of augmenting N on our parameter, α_∞. With each incremental increase in N, the relative change it induces becomes less significant. This logarithmic term encapsulates this tapering sensitivity, thereby making it a fitting choice for our regression model. Using the numerical standard errors obtained from each sample, we applied a general least square (GLS) method [29] in order to infer the values of the parameters α_∞, thereby averaging sufficient simulations to achieve a confidence interval of 99% on them. The value of α_∞ is the most important, because it is the asymptotic value that our algorithm can reach when $N \to \infty$. From the theory of GLS, we know that the estimator of the parameter α_∞ is unbiased, consistent, and efficient, and a confidence interval on this parameter is justified. The analysis, performed on data reported in Table 2, shows that the independent set ratio reached the asymptotic value $\alpha_\infty = 0.445330(3)$. This value agrees with the theoretical value proposed in [26].

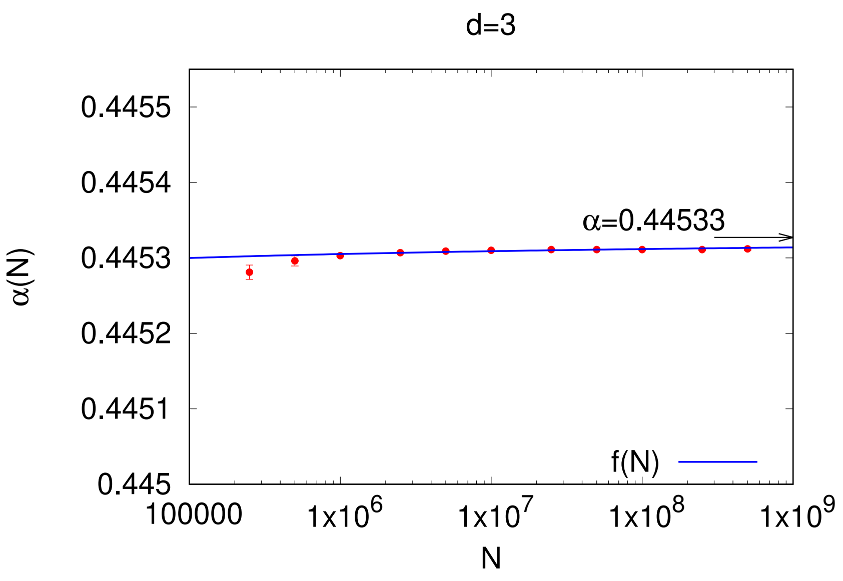

Figure 5. The figure shows the extrapolation of the independent set ratio as a function of the graph order, i.e., $|\mathcal{N}| = N$ and $d = 3$. The error bars identify the standard errors multiplied by the quantile of the t distribution $z_{99\%} = 3.35$. The asymptotic value $\alpha_\infty = 0.445330(3)$, extrapolated by fitting the data using a function $f(N) = (a/\ln N) + \alpha_\infty$ (blue line), where $a = -0.00033(6)$, identifies the values that our algorithm can reach when $N \to \infty$. The confidence interval of 99% of α_∞ (in Table 1) agrees with the theoretical value of α_{LB} = 0.44533.

Table 2. The table shows the sample average and standard deviation values of the independent set ratio $\alpha(N)$ the for random regular graphs of order N and $d = 3$.

N	d	$\alpha(N) \pm \sigma_{\alpha(N)}$
10^6	3	0.445303 (48)
$2.5 \cdot 10^6$	3	0.445307 (30)
$5 \cdot 10^6$	3	0.445309 (21)
10^7	3	0.445310 (15)
$2.5 \cdot 10^7$	3	0.445311 (9)
$5 \cdot 10^7$	3	0.445311 (7)
10^8	3	0.445311 (4)
$2.5 \cdot 10^8$	3	0.445311 (4)
$5 \cdot 10^8$	3	0.445312 (2)

4. The Deferred Decision Algorithm for $d \geq 5$

In this section, we present how to generalize the prioritized algorithm for all $d \geq 5$. It, as with the one previously described in Section 3, builds the random regular graph and, at the same time, tries to maximize the independent set cardinality $|\mathcal{I}|$. The main idea that we propose is to melt down two existing algorithms, namely, the one in [21] and the one described above, into a new prioritized algorithm, which is able to maximize the independent set cardinality, thereby providing improved estimates of the lower bounds. The new conjectured lower bounds come from extrapolation on random d-regular graphs of sizes up to 10^9.

Before introducing the algorithm, we present a new operation that will allow us to simplify the discussion.

Definition 5 ($OP^i_{build-del}(i, \mathcal{N}, \mathcal{I}, \mathcal{V})$)**.** *Let $i \in \mathcal{N}$. We define $OP^i_{build-del}(i, \mathcal{N}, \mathcal{I}, \mathcal{V})$ as the operation that connects i to $\overline{\Delta}_i$ sites following the Algorithm 1 rules, applies $OP^i_{move}(i, \mathcal{N}, \mathcal{I})$, and, $\forall j \in \partial i$, sequentially runs Algorithm 1 and applies the operation $OP^j_{move}(j, \mathcal{N}, \mathcal{V})$.*

The pseudocode of the last operation is described in Algorithm 3.

Algorithm 3: $OP^{i}_{build-del}(i,\mathcal{N},\mathcal{I},\mathcal{V})$

```
    Input: $i \in \mathcal{N}, \mathcal{N}, \mathcal{I}, \mathcal{V}$;
    Output: $\mathcal{N}, \mathcal{I}, \mathcal{V}$;
1   Run Subroutine $GA(i, \overline{\Delta}_i)$;
2   Apply $OP^{i}_{move}(i,\mathcal{N},\mathcal{I})$;
3   while $\partial i \neq \emptyset$ do
4       Pick $j \in \partial i$;
5       Run Subroutine $GA(j, \overline{\Delta}_j)$;
6       Apply $OP^{j}_{move}(j,\mathcal{N},\mathcal{V})$;
7   return $\mathcal{N}, \mathcal{I}, \mathcal{V}$;
```

We start the discussion on the local algorithm for $d \geq 5$ by giving the pseudocode of the algorithm in Algorithm 4.

The algorithm starts randomly selecting a site z from the set of all nodes $\mathcal{N}$, i.e., $z \in \mathcal{N}$. It then applies $OP^{z}_{build-del}(z,\mathcal{N},\mathcal{I},\mathcal{V})$ on the site z (see Figure 6). This operation creates nodes with different degrees and antidegrees. The algorithm proceeds in choosing the node m from those values with minimum $\overline{\Delta}_m$ values. If the node m has $\overline{\Delta}_m > 2$ values, the algorithm applies the operation $OP^{m}_{build-del}(m,\mathcal{N},\mathcal{I},\mathcal{V})$ on site m. In other words, we are using the algorithm developed in [21] until a site $m \in \mathcal{N}$ with $\overline{\Delta}_m \leq 2$ pops up. When such a case appears, we label it as a P site and we move it into the set $\mathcal{P}$.

As described in Section 3, once the set $\mathcal{P}$ is not empty, the sites in the set $\mathcal{P}$ have the highest priority in being processed for creating virtual nodes.

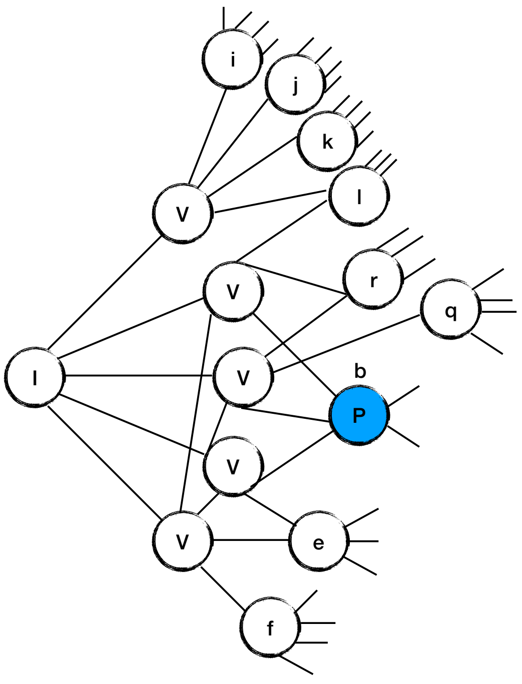

Figure 6. The figure shows how a P site appears, i.e., a site b with $\overline{\Delta}_b \leq 2$, in a random d-regular graph $G_d(\mathcal{N},E)$ with degree $d = 5$. In the event that a P site has not been created, the algorithm picks a site $m \in \mathcal{N}$ with minimum $\overline{\Delta}_m$ and applies operation $OP^{m}_{build-del}(m,\mathcal{N},\mathcal{I},\mathcal{V})$ on it.

Algorithm 4: local algorithm for $d \geq 5$

```
   Input: $N, d$;
   Output: $\mathcal{I}$;
1  Build the set of sites $\mathcal{N}$ with $|\mathcal{N}| = N$;
2  $\mathcal{I} = \emptyset$;
3  $\mathcal{V} = \emptyset$;
4  Pick a random site $i \in \mathcal{N}$ and apply $OP^{i}_{build-del}(i, \mathcal{N}, \mathcal{I}, \mathcal{V})$;
5  while $\mathcal{N} \neq \emptyset$ do
6      while $\exists i \in \mathcal{N}$ s.t. $\overline{\Delta}_i \leq 2 \wedge i \notin \mathcal{P} \wedge i$ is not labeled $C$ do
7          Label $i$ with letter $P$ and insert it into $\mathcal{P}$;
8      if $\mathcal{P} \neq \emptyset$ then
9          while $\mathcal{P} \neq \emptyset$ do
10             Pick the first $l \in \mathcal{P}$;
11             if $\overline{\Delta}_l = 0$ then
12                 Apply $OP^{l}_{move}(l, \mathcal{N}, \mathcal{I})$ and Remove $l$ from $\mathcal{P}$;
13             else
14                 Run Subroutine $GA(l, \overline{\Delta}_l)$;
15                 If a neighbour $j$ of $l$, i.e., $j \in \partial l$, is in $\mathcal{P}$ remove $j$ from $\mathcal{P}$;
16                 $\forall j \in \partial l$, label each $j$ with the letter $C$;
17                 Build or update the virtual node $\tilde{v}$ and, if it is not present, insert it
                    into $\mathcal{A}$;
18                 Remove $l$ from $\mathcal{P}$;
19     else if $\mathcal{A} \neq \emptyset$ then
20         while $\exists \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 0$ do
21             Apply $SWAP - OP(\tilde{v})$;
22             Apply $OP^{\tilde{v}}_{move}(\tilde{v}, \mathcal{A})$;
23         while $\exists \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 1$ do
24             Apply $SWAP - OP(\tilde{v})$;
25             For $i \in \tilde{v}$ labeled $P$ s.t. $\overline{\Delta}_i = 1$ run Subroutine $GA(i, \overline{\Delta}_i)$;
26             Pick $j \in \partial i$,with $j$ the last neighbour of $i$ added;
27             Run Subroutine $GA(j, \overline{\Delta}_j)$;
28             Apply $OP^{j}_{move}(j, \mathcal{N}, \mathcal{V})$;
29             Apply $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$;
30         while $\exists \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 2$ do
31             Apply $SWAP - OP(\tilde{v})$;
32             $\forall i \in \tilde{v}$ labeled $P$ s.t. $\overline{\Delta}_i \leq 2$ run Subroutine $GA(i, \overline{\Delta}_i)$ and label the
                neighbour(s) of $i$ with the letter $C$;
33             Update the virtual node $\tilde{v}$;
34         Pick $\tilde{v}$ s.t. $\max_{\tilde{v} \in \mathcal{A}} \overline{\Delta}_{\tilde{v}}$;
35         $\forall i \in \tilde{v}$ s.t. $\overline{\Delta}_i \neq 0$ and labeled $C$, run Subroutine $GA(i, \overline{\Delta}_i)$;
36         Apply $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$;
37     else
38         Pick randomly $m \in \mathcal{N}$ s.t. $\min_{m \in \mathcal{N}} \overline{\Delta}_m$ (not labeled $P$,or $C$);
39         Apply $OP^{m}_{build-del}(m, \mathcal{N}, \mathcal{I}, \mathcal{V})$;
40 return $\mathcal{I}$;
```

Until the set $\mathcal{P}$ is empty, the algorithm builds virtual sites, which are set into $\mathcal{A}$.

Once the set $\mathcal{P}$ is empty, the highest priority in being processed is placed on the virtual sites contained in $\mathcal{A}$. Again, we want the random connections outgoing from the virtual sites to reduce the antidegrees of the other existing virtual sites in $\mathcal{A}$ in such a way that the probability of having virtual sites with antidegrees of $\overline{\Delta}_{\tilde{v}} \leq 2$ becomes bigger. In order to have that, we list below the priority order that the algorithm follows for processing the virtual sites contained in $\mathcal{A}$:

1. $\forall \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 0 \vee \overline{\Delta}_{\tilde{v}} = 1$, the algorithm sequentially applies operation $SWAP - OP(\tilde{v})$ and the operation $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$. (in the case that $\overline{\Delta}_{\tilde{v}} = 1$, the algorithm applies on the last added site $j \in \partial i$ the operation $OP^{j}_{move}(j, \mathcal{N}, \mathcal{V})$ before it completes the absent connection for the site $i \in \tilde{v}$ with $\overline{\Delta}_i = 1$. Then on the virtual site $\tilde{v}$, it sequentially applies operations $SWAP - OP(\tilde{v})$ and $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$).
2. If $\exists \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} = 2 \wedge \nexists \tilde{q} \in \mathcal{A}$ s.t $\overline{\Delta}_{\tilde{q}} = 0 \vee \overline{\Delta}_{\tilde{q}} = 1$, the algorithm chooses with the highest priority the site with $\overline{\Delta}_{\tilde{v}} = 2$. Then, it applies operation $SWAP - OP(\tilde{v})$ on $\tilde{v}$, it runs the Subroutine $GA(i, \overline{\Delta}_i)$ $\forall i \in \tilde{v}$ with $\overline{\Delta}_i \neq 0$, and it labels each neighbor(s) of i with letter C.
3. If $\exists \tilde{v} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{v}} > 2 \wedge \nexists \tilde{p} \in \mathcal{A}$ s.t. $\overline{\Delta}_{\tilde{p}} \leq 2$, the algorithm chooses a site $\tilde{v} \in \mathcal{A}$ with the maximum $\overline{\Delta}_{\tilde{v}}$, and it applies the $OP^{\tilde{v}}_{del}(\tilde{v}, \mathcal{A})$ with the maximum $\overline{\Delta}_{\tilde{v}}$ after having run the Subroutine $GA(i, \overline{\Delta}_i)$ on each $i \in \tilde{v}$ such that $\overline{\Delta}_i \neq 0$.
4. In the case that $\mathcal{P} = \emptyset \wedge \mathcal{A} = \emptyset \wedge \mathcal{N} \neq \emptyset$, the algorithm takes a site $t \in \mathcal{N}$ with a minimum $\overline{\Delta}_t$, and it applies the operation $OP^{t}_{build-del}(t, \mathcal{N}, \mathcal{I}, \mathcal{V})$.

The algorithm works until the following condition is true: $\mathcal{N} = \emptyset \wedge \mathcal{P} = \emptyset \wedge \mathcal{A} = \emptyset$. Then, it checks that all sites in $\mathcal{I}$ are covered only by sites in $\mathcal{V}$, and it checks that no site in $\mathcal{I}$ connects to any other site in $\mathcal{I}$.

The results obtained by the algorithm for different values of d, and different orders N, are presented in Tables 3–5 . The confidence intervals of the asymptotic independent set ratio values, obtained using the extrapolation described in the previous section, are presented in Table 1. In other words, we performed simulations for each value of d by computing the sample mean and the standard error of the independence ratio for some values of N. Then, we used GLS methods in order to extrapolate the values of α_∞ and build up its confidence interval.

From our analysis, we observed that, $\forall d > 4$, our results, as far as we know, exceed the best theoretical lower bounds given by greedy algorithms. Those improvements were obtained because we allowed the virtual nodes to increase and decrease their antidegrees. In other words, this process transforms the random d-regular graph into a sparse random graph, wherein it is much easier making local rearrangements (our $SWAP - OP(\cdot)$ move) to enlarge the independent set. More precisely, the creation of virtual nodes that increase or decrease their antidegrees allows us to deal with a graph that is no longer d-regular but has average connectivity $\langle d \rangle$.

Table 3. The table shows the sample average and standard deviation values of the independent set ratio $\alpha(N)$ for random regular graphs of order N and degree $d = 5, 6, 7$.

N	d	$\alpha(N) \pm \sigma_{\alpha(N)}$	d	$\alpha(N) \pm \sigma_{\alpha(N)}$	d	$\alpha(N) \pm \sigma_{\alpha(N)}$
10^6	5	0.364723 (78)	6	0.335964 (84)	7	0.312367 (89)
$2.5 \cdot 10^6$	5	0.364731 (48)	6	0.335969 (53)	7	0.312373 (56)
$5 \cdot 10^6$	5	0.364732 (35)	6	0.335972 (38)	7	0.312378 (37)
10^7	5	0.364733 (24)	6	0.335975 (26)	7	0.312378 (27)
$2.5 \cdot 10^7$	5	0.364734 (15)	6	0.335976 (18)	7	0.312380 (18)
$5 \cdot 10^7$	5	0.364735 (11)	6	0.335975 (12)	7	0.312381 (13)
10^8	5	0.364734 (7)	6	0.335975 (8)	7	0.312381 (9)
$2.5 \cdot 10^8$	5	0.364735 (5)	6	0.335975 (6)	7	0.312381 (5)
$5 \cdot 10^8$	5	0.364734 (3)	6	0.335977 (2)	7	0.312381 (4)

Table 4. The table shows the sample average and standard deviation values of the independent set ratio $\alpha(N)$ for random regular graphs of order N and degree $d = 8, 9, 10$.

N	d	$\alpha(N) \pm \sigma_{\alpha(N)}$	d	$\alpha(N) \pm \sigma_{\alpha(N)}$	d	$\alpha(N) \pm \sigma_{\alpha(N)}$
10^6	8	0.292522 (83)	9	0.275511 (85)	10	0.260747 (84)
$2.5 \cdot 10^6$	8	0.292523 (53)	9	0.275517 (53)	10	0.260753 (54)
$5 \cdot 10^6$	8	0.292526 (37)	9	0.275519 (38)	10	0.260755 (38)
10^7	8	0.292527 (26)	9	0.275521 (27)	10	0.260757 (28)
$2.5 \cdot 10^7$	8	0.292529 (17)	9	0.275522 (17)	10	0.260759 (17)
$5 \cdot 10^7$	8	0.292530 (11)	9	0.275521 (12)	10	0.260758 (11)
10^8	8	0.292530 (8)	9	0.275523 (8)	10	0.260759 (8)
$2.5 \cdot 10^8$	8	0.292530 (4)	9	0.275523 (5)	10	0.260759 (5)

Table 5. The table shows the sample average and standard deviation values of the independent set ratio $\alpha(N)$ for random regular graphs of order N and degree $d = 20$, 50, 100.

N	d	$\alpha(N) \pm \sigma_{\alpha(N)}$	d	$\alpha(N) \pm \sigma_{\alpha(N)}$	d	$\alpha(N) \pm \sigma_{\alpha(N)}$
$2.5 \cdot 10^5$	20	0.175389 (151)	50	0.095673 (114)	100	0.057523 (88)
$5 \cdot 10^5$	20	0.175403 (107)	50	0.095682 (81)	100	0.057522 (62)
10^6	20	0.175407 (75)	50	0.095684 (57)	100	0.057524 (43)
$2.5 \cdot 10^6$	20	0.175412 (48)	50	0.095688 (36)	100	0.057525 (27)
$5 \cdot 10^6$	20	0.175414 (33)	50	0.095689 (24)	100	0.057527 (20)
10^7	20	0.175415 (24)	50	0.095690 (18)	100	0.057528 (14)
$2.5 \cdot 10^7$	20	0.175416 (17)	50	0.095691 (12)	100	0.057527 (9)
$5 \cdot 10^7$	20	0.175418 (11)	50	0.095690 (9)	100	0.057527 (7)

However, this improvement decreases as d becomes large, $\sim 1/d$, and disappears when $d \to \infty$ (see Figure 7, bottom panel). Indeed, the number of P labeled sites decreased during the graph-building process (see Figure 7, top panel), thus invalidating the creation of the virtual nodes that are at the core of our algorithm. This means that our algorithm for values of d, such that $d \to \infty$, will reach the same asymptotic independent set ratio values obtained by the algorithm in [21].

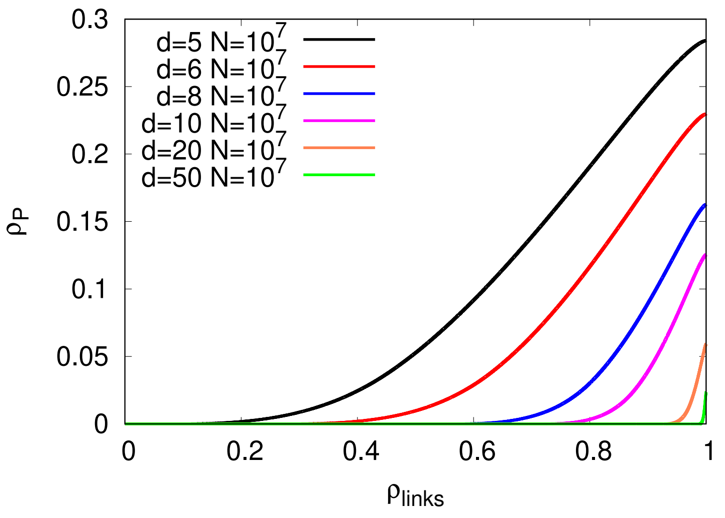

Figure 7. *Cont.*

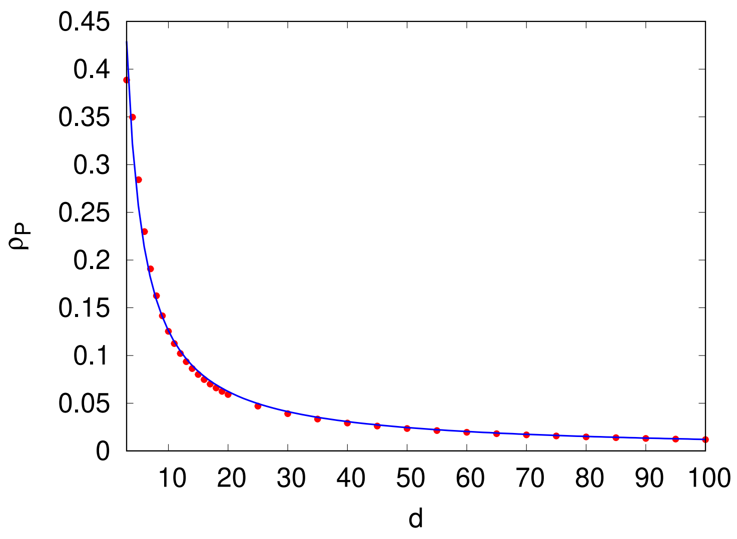

Figure 7. The figure shows how the dynamics of the fraction of P sites in the graph-building process appear as a function of links inserted in (**top panel**), and the total fraction of P sites appears as a function of d (**lower panel**). The fraction of P sites decreases from $\sim 1/d$ to when d becomes large. As stated in the main text, this behavior shows that, for $d \to \infty$, our algorithm matches the one in [21].

In conclusion, for any fixed and small d, we have that the two algorithms are distinct, and our algorithm produces better results without increasing the computational complexity.

5. Conclusions

This manuscript presents a new local prioritized algorithm for finding a large independent set in a random d-regular graph at fixed connectivity. This algorithm makes a deferred decision in choosing which site must be set into the independent set or into the vertex cover set. This deferred strategy can be seen as a depth-first search delayed in time, without backtracking. It works, and it shows very interesting results.

For all $d \in [5, 100]$, we conjecture new lower bounds for this problem. All the new bounds improve upon the best previous bounds. All of them have been obtained using extrapolation on the samples of random d-regular graphs of sizes up to 10^9. For random 3-regular graphs, our algorithm is able to reach, when $N \to \infty$, the asymptotic value presented in [26]. We recall that the algorithm in [26] cannot be used for any $d > 4$. However, we think that this approach can also provide conjectured lower bound for $d > 100$. Indeed, as shown in Figure 7, there is still space to use our algorithm for computing independent sets. Moreover, new strategies could be implemented for improving our conjectured bounds.

The improvements upon the best bounds are due to reducing the density of the graph, thereby introducing regions in which virtual sites replace multiple original nodes, and optimal labelings can be identified. The creation of virtual sites makes it possible to group together nodes of the graph to label each at a different instant with respect to their creation. Those blobs of nodes transform the random d-regular graphs into a sparse graph, where the searching of a large independent set is simpler.

Undoubtedly, more complex virtual nodes can be defined, and additional optimizations can be identified. This will be addressed in a future manuscript.

Author Contributions: Conceptualization, R.M. and S.K.; methodology, R.M. and S.K.; software, R.M.; validation, R.M. and S.K.; formal analysis, R.M. and S.K.; investigation, R.M. and S.K.; resources, R.M. and S.K.; data curation, R.M. and S.K.; writing—original draft preparation, R.M. and S.K.; writing—review and editing, R.M. and S.K.; visualization, R.M. and S.K.; supervision, R.M. and S.K.; project administration, R.M. and S.K.; All authors have read and agreed to the published version of the manuscript.

Funding: S.K. is supported by the Federman Cyber Security Center at the Hebrew University of Jerusalem. R.M. is supported by #NEXTGENERATIONEU (NGEU) and funded by the Ministry of University and Research (MUR), the National Recovery and Resilience Plan (NRRP), and the project MNESYS (PE0000006)—A multiscale integrated approach to the study of the nervous system in health and disease (DN. 1553 11.10.2022).

Data Availability Statement: The code of the algorithm can be released upon request.

Acknowledgments: R.M. would like to thank Nicolas Macris for a first reading of the manuscript and Endre Csóka for useful discussions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cook, S. The P versus NP problem. In *The Millennium Prize Problems*; American Mathematical Society: Providence, RI, USA, 2006; pp. 87–104.
2. Tarjan, R.E.; Trojanowski, A.E. Finding a maximum independent set. *SIAM J. Comput.* **1977**, *6*, 537–546. [CrossRef]
3. Xiao, M.; Nagamochi, H. Exact algorithms for maximum independent set. *Inf. Comput.* **2017**, *255*, 126–146. [CrossRef]
4. Mezard, M.; Montanari, A. *Information, Physics, and Computation*; Oxford University Press: Oxford, UK, 2009.
5. Mohseni, M.; Eppens, D.; Strumpfer, J.; Marino, R.; Denchev, V.; Ho, A.K.; Isakov, S.V.; Boixo, S.; Ricci-Tersenghi, F.; Neven, H. Nonequilibrium Monte Carlo for unfreezing variables in hard combinatorial optimization. *arXiv* **2021**, arXiv:2111.13628.
6. Marino, R.; Parisi, G.; Ricci-Tersenghi, F. The backtracking survey propagation algorithm for solving random K-SAT problems. *Nat. Commun.* **2016**, *7*, 12996. [CrossRef] [PubMed]
7. Marino, R.; Macris, N. Solving non-linear Kolmogorov equations in large dimensions by using deep learning: A numerical comparison of discretization schemes. *J. Sci. Comput.* **2023**, *94*, 8. [CrossRef]
8. Marino, R. Learning from survey propagation: A neural network for MAX-E-3-SAT. *Mach. Learn. Sci. Technol.* **2021**, *2*, 035032. [CrossRef]
9. Bomze, I.M.; Budinich, M.; Pardalos, P.M.; Pelillo, M. The maximum clique problem. In *Handbook of Combinatorial Optimization*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 1–74.
10. Karp, R.M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*; Springer: Berlin/Heidelberg, Germany, 1972; pp. 85–103.
11. Marino, R.; Kirkpatrick, S. Hard optimization problems have soft edges. *Sci. Rep.* **2023**, *13*, 3671. [CrossRef]
12. Wein, A.S. Optimal Low-Degree Hardness of Maximum Independent Set. *arXiv* **2020**, arXiv:2010.06563.
13. Gamarnik, D.; Sudan, M. Limits of local algorithms over sparse random graphs. In Proceedings of the 5th Conference on Innovations in Theoretical Computer Science, Princeton, NJ, USA, 12–14 January 2014; pp. 369–376.
14. Rahman, M.; Virag, B. Local algorithms for independent sets are half-optimal. *Ann. Probab.* **2017**, *45*, 1543–1577. [CrossRef]
15. Ding, J.; Sly, A.; Sun, N. Maximum independent sets on random regular graphs. *Acta Math.* **2016**, *217*, 263–340. [CrossRef]
16. Bollobás, B. The independence ratio of regular graphs. *Proc. Am. Math. Soc.* **1981**, *83*, 433–436. [CrossRef]
17. McKay, B. Independent sets in regular graphs of high girth. *Ars Comb.* **1987**, *23*, 179–185.
18. Balogh, J.; Kostochka, A.; Liu, X. Cubic graphs with small independence ratio. *arXiv* **2017**, arXiv:1708.03996.
19. Barbier, J.; Krzakala, F.; Zdeborová, L.; Zhang, P. The hard-core model on random graphs revisited. In Proceedings of the ELC International Meeting on Inference, Computation, and Spin Glasses (ICSG2013), Sapporo, Japan, 28–30 July 2013; Journal of Physics: Conference Series; IOP Publishing: Bristol, UK, 2013; Volume 473, p. 012021.
20. Harangi, V. Improved replica bounds for the independence ratio of random regular graphs. *J. Stat. Phys.* **2023**, *190*, 60. [CrossRef]
21. Wormald, N.C. Differential equations for random processes and random graphs. *Ann. Appl. Probab.* **1995**, *5*, 1217–1235. [CrossRef]
22. Shearer, J.B. A note on the independence number of triangle-free graphs. *Discret. Math.* **1983**, *46*, 83–87. [CrossRef]
23. Wormald, N.C. Analysis of greedy algorithms on graphs with bounded degrees. *Discret. Math.* **2003**, *273*, 235–260. [CrossRef]
24. Duckworth, W.; Zito, M. Large independent sets in random regular graphs. *Theor. Comput. Sci.* **2009**, *410*, 5236–5243. [CrossRef]
25. Csóka, E.; Gerencsér, B.; Harangi, V.; Virág, B. Invariant Gaussian processes and independent sets on regular graphs of large girth. *Random Struct. Algorithms* **2015**, *47*, 284–303. [CrossRef]
26. Csóka, E. Independent sets and cuts in large-girth regular graphs. *arXiv* **2016**, arXiv:1602.02747.
27. Angelini, M.C.; Ricci-Tersenghi, F. Monte Carlo algorithms are very effective in finding the largest independent set in sparse random graphs. *Phys. Rev. E* **2019**, *100*, 013302. [CrossRef] [PubMed]
28. Hoppen, C.; Wormald, N. Local algorithms, regular graphs of large girth, and random regular graphs. *Combinatorica* **2018**, *38*, 619–664. [CrossRef]
29. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.