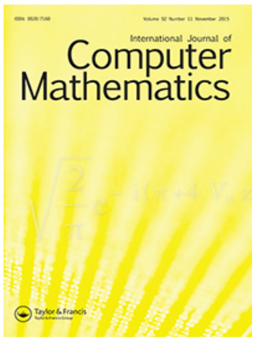

International Journal of Computer Mathematics

ISSN: 0020-7160 (Print) 1029-0265 (Online) Journal homepage: http://www.tandfonline.com/loi/gcom20

On the independent set problem in random graphs

Yinglei Song

To cite this article: Yinglei Song (2015) On the independent set problem in random graphs, International Journal of Computer Mathematics, 92:11, 2233-2242, DOI: 10.1080/00207160.2014.976210

To link to this article: http://dx.doi.org/10.1080/00207160.2014.976210

Accepted author version posted online: 14 Oct 2014.
Published online: 06 Nov 2014.

International Journal of Computer Mathematics, 2015
Vol. 92, No. 11, 2233–2242, http://dx.doi.org/10.1080/00207160.2014.976210

On the independent set problem in random graphs

Yinglei Song*

School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu 212003, China

(*Received 28 March 2013; revised version received 3 August 2014; accepted 8 October 2014*)

In this paper, we develop efficient exact and approximate algorithms for computing a maximum independent set in random graphs. In a random graph G, each pair of vertices are joined by an edge with a probability p, where p is a constant between 0 and 1. We show that a maximum independent set in a random graph that contains n vertices can be computed in expected computation time $2^{O(\log_2^2 n)}$. In addition, we show that, with high probability, the parameterized independent set problem is fixed parameter tractable in random graphs and the maximum independent set in a random graph in n vertices can be approximated within a ratio of $2n/2^{\sqrt{\log_2 n}}$ in expected polynomial time.

Keywords: independent set; random graphs; exact algorithm; parameterized algorithm; approximate algorithm

2010 AMS Subject Classification: 05C85

1. Introduction

In computer science, many optimization problems can be reduced to the optimization of objectives that are formulated and described in a graph. The development of efficient exact or approximate algorithms for graph optimization problems thus constitutes an important part of the research in combinatorial optimization. However, a large number of graph optimization problems have been shown to be NP-hard [14], which suggests that it is unlikely to develop algorithms that can solve these problems in polynomial time. A well-known example is the Maximum Independent Set problem. Given a graph $G = (V, E)$, a vertex set $I \subseteq V$ is an *independent set* if there is no edge between any pair of two vertices in I. The goal of the Maximum Independent Set problem is to find an independent set of the largest size in a given graph G. If G contains n vertices in total, the problem can be trivially solved in time $2^{O(n)}$ by enumerating and checking all possible vertex subsets in the graph. Although intensive research has been performed to improve the computation time needed to find an optimal solution [13,19,23,25], an algorithm that needs subexponential time is not yet available for this problem. Recently, it is proposed that this problem is unlikely to be solved in subexponential time [4,5].

Due to the difficulty in developing efficient algorithms that can find optimal solutions for these problems, a large number of algorithms have been developed to obtain approximate solutions for these problems in a significantly reduced amount of computation time [20]. These algorithms

*Email: yingleisong@gmail.edu

can often achieve a trade-off between the optimality of the solutions and the computation time needed to obtain them. For example, a simple algorithm that computes a maximal matching in a graph can approximate its minimum vertex cover within a ratio of 2.0. For some NP-hard problems, approximate solutions that are within a certain approximate ratio cannot be obtained in polynomial time unless NP = P. As an example, it has been shown that it is NP-hard to approximate the minimum vertex cover in a graph within a ratio of 1.362 [6]. Another well-known inapproximability result regarding the Maximum Independent Set problem is that it is NP-hard to approximate the maximum independent set in a graph within a ratio of $n^{1-\epsilon}$, where $0 < \epsilon < 1$ is a constant and n is the number of vertices in the graph [17]. This result suggests that an approximate solution with a guaranteed constant approximate ratio cannot be obtained in polynomial time for the Maximum Independent Set problem unless NP = P. So far, the best known approximation ratio that has been achieved for this problem in general graphs is $O(n \log_2^2 \log_2 n / \log_2^3 n)$ [11].

For those problems that cannot be even approximated within a good approximation ratio in polynomial time, such as the Maximum Independent Set problem, heuristics that can efficiently generate approximate solutions are often employed in practice to solve them [2,16,22]. However, solutions generated by heuristics are not guaranteed to be close to the optimal ones and their applications are thus restricted to scenarios where the accuracy of solutions is not a crucial issue.

Parameterized computation is an alternative approach that may lead to practically efficient algorithms for some NP-hard problems. In practice, an instance of an NP-hard problem may contain one or a few parameters, parameterized computation focuses on the development of exact algorithms that can efficiently solve the problem while these parameters are small positive integers. Specifically, if we denote these parameters by $p_1, p_2, \ldots, p_t$, the problem is *fixed parameter tractable* if there exists an algorithm that can solve the problem in time $O(f(p_1, p_2, \ldots, p_t)n^d)$, where n is the size of the problem, f is a function of the parameters and d is a constant that does not depend on n or any of the parameters $p_1, p_2, \ldots, p_t$. A well-known example of fixed parameter tractable problems is Vertex Cover. In [8], it is shown that there exists an algorithm that can determine whether a graph contains a vertex cover of size k or not in time $O(2^k n)$, where n is the number of vertices in the graph. However, not all NP-hard problems are known to be fixed parameter tractable and it has been shown that some of them are unlikely to be solved by efficient parameterized algorithms. Different parameterized complexity classes have been developed in parameterized complexity theory to reflect the parameterized intractability of these problems [8]. For example, the Independent Set problem is known to be complete for complexity class W[1] [9], it thus cannot be solved by an efficient parameterized algorithm unless W[1] collapses into the class of fixed parameter tractable problems. A comprehensive survey on parameterized algorithms and parameterized complexity theory can be found in [7].

In this paper, we develop exact and approximate algorithms for the Maximum Independent Set problem where the underlying graph is a random graph generated based on the Erdős–Rényi model [10]. Such a random graph is generated by treating each pair of vertices independently and adding an edge to join them with a probability of p $(0 < p < 1)$, where p is a constant. Recent research in molecular biology has shown that the protein side chain interaction network conforms remarkably well to random graphs generated by the Erdős–Rényi model [24]. Therefore, efficient algorithms for some NP-hard problems in random graphs, if exist, may significantly improve the computational efficiency for some important optimization problems related to protein structure prediction.

In [15,21], it has been shown that with high probability, the maximum independent set in a random graph is of size $O(\log_2 n)$. However, this result does not directly lead to an algorithm that can compute the maximum independent set in a random graph in expected subexponential time. In [12], a polynomial time algorithm that can compute a maximum independent set in a sparse random graph with high probability is developed. However, the algorithm is based on a large independent set that is embedded in the graph and thus cannot be used for all graphs. We show that the maximum independent set in a random graph can be computed in expected computation time $2^{O(\log_2^2 n)}$, where n is the number of vertices in the graph. This result significantly improves the best known time complexity $O(2^{n/4})$ for finding a maximum independent set in general graphs [25].

In addition, we show that, with high probability, the parameterized independent set problem is fixed parameter tractable in random graphs. For approximate algorithms, we develop an algorithm that can achieve an approximation ratio of $2n/2^{\sqrt{\log_2 n}}$ in expected polynomial time, which is a significant improvement compared with the best known approximate ratio that can be achieved in general graphs [11].

2. Maximum independent set in random graphs

A *random graph* $G(V,p)$, where $0 < p < 1$, is a graph obtained by independently adding edges between each pair of vertices in V with a probability p. Given a vertex $v \in V$, the *degree* of v in G is the number of vertices that are connected to v by an edge in G. We use $\deg_G(v)$ to denote the degree of vertex v in graph G and $N_G(v)$ to denote the set of vertices that are connected to v by an edge in G. A vertex subset $I \subseteq V$ is an independent set in G if there is no edge between any pair of vertices in I. The goal of the MAXIMUM INDEPENDENT SET problem is to find an independent set of the largest size in a given graph.

In [15,21], it is shown that, with high probability, the size of a maximum independent set in a random graph $G(V,p)$ is $2\log_2 n/\log_2 1/(1-p)$, where n is the number of vertices in G. A straightforward algorithm by exhaustively enumerating all vertex subsets of size $2\log_2 n/\log_2 1/(1-p)$ can thus compute a maximum independent set in most random graphs in time $n^{O(\log_2 n)}$. However, to compute a maximum independent set in all random graphs, the algorithm must be able to cope with the cases where the graph contains an independent set of size larger than $O(\log_2 n)$. The algorithm needs time $2^{O(n)}$ to compute a maximum independent set in these cases. The best known upper bound of the probability for a random graph to has a maximum independent set larger than $O(\log_2 n)$ is $1/n^{O(1)}$ [15,21], the expected time complexity of this enumeration based algorithm is thus $2^{O(n)}$.

We show that the maximum independent set in a random graph $G = (V,p)$ can be computed in expected subexponential time.

LEMMA 2.1 *Given a random graph $G = (V,p)$ where $n = |V|$ and a sufficiently small constant ϵ such that $\epsilon < p$, there exists a vertex $v \in V$ such that* $\deg_G(v) \geq (p-\epsilon)n$ *with probability at least* $1 - 2^{-\mu n^2}$, *where μ is a positive constant that only depends on ϵ and p.*

Proof If such a vertex does not exist, the number of edges $n(E)$ in G is at most $(p-\epsilon)n^2/2$ since the degree of each vertex is at most $(p-\epsilon)n$. However, from the construction of graph G, the expected number of edges in G can be obtained as follows:

$$E(n(E)) = \frac{pn(n-1)}{2}. \tag{1}$$

From Chernoff bound, we can bound the probability for $n(E) < (p-\epsilon)n^2/2$ by

$$\Pr\left(n(E) < \frac{(p-\epsilon)n^2}{2}\right) < \exp\left(-\frac{pn(n-1)\delta^2}{4}\right), \tag{2}$$

where $\delta = (n\epsilon - p)/p(n-1)$. For sufficiently large n, we have

$$\delta > \frac{\epsilon}{2p}, \tag{3}$$

$$n - 1 > \frac{n}{2}. \tag{4}$$

We can thus immediately obtain

$$\Pr\left(n(E) < \frac{(p+\epsilon)n^2}{2}\right) \tag{5}$$

$$< \exp\left(-\frac{\epsilon^2 n^2}{32p}\right) \tag{6}$$

$$= 2^{-\epsilon^2 n^2/32p\ln 2}. \tag{7}$$

We then let $\mu = \epsilon^2/32p\ln 2$ and we conclude that with probability at least $1 - 2^{-\mu n^2}$, there exists vertex $v \in V$ such that $\deg_G(v) \geq (p-\epsilon)n$. ■

The proof of Lemma 2.1 relies on the fact that p is a constant independent of n, the lemma does not hold if the value of p depends on n. A random graph $G = (V, p)$ in n vertices is *good* if it contains at least one vertex whose degree is at least $(p-\epsilon)n$. Given a random graph, the algorithm starts by finding a vertex v such that $\deg_G(v)$ is at least $(p-\epsilon)n$. If such a vertex does not exist, the algorithm enumerates all subsets of V and returns an independent set of the largest size. If v exists, the algorithm branches on two possible cases on whether v is contained in I or not. In particular, if $v \in I$, v and vertices in $N(v)$ are deleted from G and the resulting graph is G_1; if $v \notin I$, v is deleted from G and the resulting graph is G_2. The algorithm is then recursively applied on both G_1 and G_2 to compute a maximum independent set in each of them. We use I_1 and I_2 to denote the maximum independent sets in G_1 and G_2 found by the algorithm, respectively. I_2 is returned as a maximum independent set in G if $|I_2| \geq |I_1| + 1$ and $I_1 \cup \{v\}$ is returned otherwise. We show that this algorithm terminates in expected time $2^{O(\log_2^2 n)}$.

THEOREM 2.1 *A maximum independent set in a random graph $G = (V, p)$ with n vertices can be computed in expected computation time $2^{O(\log_2^2 n)}$.*

Proof We show that the algorithm described above terminates in expected time $2^{O(\log_2^2 n)}$. In particular, the algorithm is recursive and for each step of recursion, we have the following recursion relation for the computation time if the underlying graph is good and contains m vertices

$$T(m) \leq T((1-p+\epsilon)m) + T(m-1) + O(m^2), \tag{8}$$

where $T(m)$ is the computation time needed by the algorithm in a graph on m vertices. The term $O(m^2)$ is the computation time needed to find a vertex whose degree is at least $(p-\epsilon)m$, since the time needed to compute the degree of a vertex is $O(m)$ and the algorithm may need to check m vertices to find such a vertex. If the underlying graph is not good, the algorithm exhaustively enumerates all subsets in the graph and finds an independent set of the largest size. The computation time is $2^{O(m)}$.

We are now ready to establish the expected computation time for the algorithm. In particular, we use $\mathrm{ET}(m)$ to denote the expected computation time of the algorithm on a graph that contains

m vertices. From Lemma 2.1, an underlying graph G' in m vertices is good with a probability of at least $1 - 2^{-\mu m^2}$. We thus can immediately obtain the following recursion for $\mathrm{ET}(m)$:

$$\mathrm{ET}(m) \leq \mathrm{ET}((1-p+\epsilon)m) + \mathrm{ET}(m-1) + O(m^2) + 2^{O(m)-\mu m^2} \tag{9}$$

$$\leq \mathrm{ET}((1-p+\epsilon)m) + \mathrm{ET}(m-1) + O(m^2), \tag{10}$$

where the second inequality is due to the fact that $2^{O(m)-\mu m^2}$ is bounded by a constant for all positive integers m.

We then show that $\mathrm{ET}(m) \leq 2^{c\log_2^2 m}$, where c is a positive constant. We show this by induction. First, for a sufficiently large positive integer m_0 whose value will be specified later, we let $c_0 = \max_{1\leq t\leq m_0}\{\log_2 \mathrm{ET}(t)/\log_2^2 t\}$ and choose $c = \max\{c_0, 2/\log_2(1/(1-p+\epsilon)), 1\}$. It is not difficult to see that $\mathrm{ET}(l) \leq 2^{c\log_2^2 l}$ if $1 \leq l \leq m_0$. We then assume that this holds for all positive integers less than m. From the above recursion relation on $\mathrm{ET}(m)$, we can obtain

$$\mathrm{ET}(m) \leq 2^{c\log_2^2((1-p+\epsilon)m)} + 2^{c\log_2^2(m-1)} + Bm^2 \tag{11}$$

$$\leq sm^{-l}2^{c\log_2^2 m} + 2^{c\log_2^2 m} + (2^{c\log_2^2(m-1)} - 2^{c\log_2^2 m}) + Bm^2 \tag{12}$$

$$\leq sm^{-l}2^{c\log_2^2 m} + 2^{c\log_2^2 m} - \frac{\log_2 m}{24m}2^{c\log_2^2 m} + Bm^2 \tag{13}$$

$$\leq 2^{c\log_2^2 m} \tag{14}$$

where B is a positive constant independent of c, p, ϵ and s, q, l are some positive constants that depend on c, p, ϵ only. The first inequality is obtained from the assumption for induction. The second one is due to the fact that $\log_2^2((1-p+\epsilon)m) = \log_2^2(1-p+\epsilon) + 2\log_2(1-p+\epsilon)\log_2 m + \log_2^2 m$ and we can let $l = 2c\log_2 1/(1-p+\epsilon)$, $s = 2^{c\log_2^2(1-p+\epsilon)}$.

To establish the third inequality, we have

$$\log_2^2(m-1) - \log_2^2 m = \left(\log_2 m + \log_2\left(1 - \frac{1}{m}\right)\right)^2 - \log_2^2 m \tag{15}$$

$$\leq \left(\log_2 m - \frac{1}{6m}\right)^2 - \log_2^2 m \tag{16}$$

$$\leq -\frac{\log_2 m}{6m} \tag{17}$$

$$\leq -\frac{\log_2 m}{6cm}, \tag{18}$$

when $m \geq 16$, we can obtain

$$2^{c\log_2^2(m-1)} - 2^{c\log_2^2 m} = 2^{c\log_2^2 m}(2^{c(\log_2^2(m-1)-\log_2^2 m)} - 1) \tag{19}$$

$$\leq 2^{c\log_2^2 m}(2^{-\log_2 m/6m} - 1) \tag{20}$$

$$\leq -\frac{\log_2 m}{24m}2^{c\log_2^2 m}, \tag{21}$$

the third inequality thus follows.

From the fact that $c \geq 2/\log_2 1/(1-p+\epsilon)$, we have $l \geq 4$. We let

$$c' = \frac{2}{\log_2 1/(1-p+\epsilon)}, \tag{22}$$

$$s' = 2^{c' \log_2^2((1-p+\epsilon)m)}, \tag{23}$$

$$l' = 2c' \log_2 \frac{1}{1-p+\epsilon}, \tag{24}$$

we now consider the function $F(m) = (s'm^{-l'} - \log_2 m/24m)2^{c' \log_2^2 m} + Bm^2$. Since s', l', c', and B are independent of m and $l' \geq 4$, there exists a positive integer $m_1(p,\epsilon)$ such that $F(m) \leq 0$ when $m \geq m_1(p,\epsilon)$. m_0 can be determined as follows:

$$m_0 = \max\left\{m_1(p,\epsilon), \frac{1}{\sqrt{1-p+\epsilon}},\ 16\right\}. \tag{25}$$

It is not difficult to see that when $c \geq c'$ and $m \geq m_0$, we have $s'm^{-l'} - \log_2 m/24m \leq 0$. In addition, we can further verify that

$$sm^{-l} = 2^{c \log_2(1-p+\epsilon)\log_2(m^2(1-p+\epsilon))}, \tag{26}$$

since $c \geq c'$, $m \geq 1/\sqrt{1-p+\epsilon}$, and $\log_2(1-p+\epsilon) \leq 0$, we can immediately obtain

$$sm^{-l} = 2^{c \log_2(1-p+\epsilon)\log_2(m^2(1-p+\epsilon))} \tag{27}$$

$$\leq 2^{c' \log_2(1-p+\epsilon)\log_2(m^2(1-p+\epsilon))} \tag{28}$$

$$= s'm^{-l'} \tag{29}$$

the following thus holds

$$\left(sm^{-l} - \frac{\log_2 m}{24m}\right)2^{c\log_2^2 m} + Bm^2 \leq \left(s'm^{-l'} - \frac{\log_2 m}{24m}\right)2^{c\log_2^2 m} + Bm^2 \tag{30}$$

$$\leq \left(s'm^{-l'} - \frac{\log_2 m}{24m}\right)2^{c'\log_2^2 m} + Bm^2 \tag{31}$$

$$= F(m) \tag{32}$$

$$\leq 0, \tag{33}$$

the fourth inequality thus follows. From the principle of induction, the theorem has been proved. ■

3. Parameterized algorithm for independent set problem

The parameterized independent set problem is to decide whether a given graph $G = (V,E)$ contains an independent set of size k or not. The problem is known to be W[1]-hard [7–9] and cannot be solved in time $n^{o(k)}$ in general graphs unless W[2] = FPT [4,5]. We show that if the underlying graph G is a random graph, the problem can be solved in expected time $2^{O(k^2)} + O(n^3)$, where n is the number of vertices in the graph. We need the following lemma to analyse the time complexity of the algorithm.

LEMMA 3.1 *Given a random graph $G = (V, p)$ where $n = |V|$ and a sufficiently small constant ϵ such that $p + \epsilon < 1$, there exists vertex $u \in V$ such that $\deg_G(u) \leq (p + \epsilon)n$ with a probability of at least $1 - 2^{-\mu n^2}$, where μ is a positive constant that only depends on ϵ and p.*

Proof The proof is similar to the proof of Lemma 2.1. If such a vertex does not exist, the degree of every vertex in G is at least $(p + \epsilon)n$. The graph thus contains at least $(p + \epsilon)n^2/2$ edges. The expected number of edges in G is $pn(n - 1)/2$. We use $n(E)$ to denote the number of the edges in G. From Chernoff bound, we can bound the probability for G to contain at least $(p + \epsilon)n^2/2$ edges

$$\Pr\left(n(E) \geq \frac{(p+\epsilon)n^2}{2}\right) \tag{34}$$

$$< \exp\left(-\frac{\epsilon^2 n^2}{64p}\right) \tag{35}$$

$$= 2^{-\epsilon^2 n^2/64p \ln 2} \tag{36}$$

the lemma immediately follows by letting $\mu = \epsilon^2/64p \ln 2$. ■

The proof of Lemma 3.1 relies on the fact that p is a constant independent of n, the lemma does not hold if the value of p depends on n.

THEOREM 3.1 *Given a random graph $G = (V, p)$, there exists an algorithm that can decide whether G contains an independent set of size k in expected time $2^{O(k^2)} + O(n^3)$.*

Proof We start the proof by comparing the values of k and $L(n) = (1/3)\log_{1/(1-p-\epsilon)} n$, if $k > L(n)$, we can enumerate all possible vertex subsets of size k in G and check whether one of them is an independent set of size k or not. The enumeration and checking needs at most $O(k^2 n^k)$ time. However, since $k > L(n)$, we can obtain $n < (1/(1 - p - \epsilon))^{3k}$, the computation time needed to determine whether G contains an independent set of size k or not is thus at most $O(k^2(1/(1 - p - \epsilon))^{3k^2}) = 2^{O(k^2)}$ in this case.

We then consider the case where $k \leq L(n)$. We use the following procedure to generate an independent set I. We start with the vertex u with the minimum degree in G, we include u in I and remove u and all its neighbors in G from G. We denote the resulting graph by G_1. The procedure can be repeatedly executed until there are at most $n^{2/3}$ vertices left in the graph. We use $G_0 = G, G_1, G_2, G_3, \ldots, G_l$ to denote the intermediate graphs generated during this iterative procedure. It is not difficult to see that vertices in I form an independent set in G.

We show that the above procedure can generate an independent set I of size at least $L(n)$ with high probability. We use $G_1, G_2, G_3, \ldots, G_l$ to denote the resulting graph in each iterative step and $n(G_i)$ to denote the number of vertices in graph G_i. From Lemma 3.1, the following holds with a probability of at least $1 - 2^{-\mu n^2(G_i)}$ for each i between 0 and l.

$$n(G_{i+1}) \geq (1 - p - \epsilon)n(G_i). \tag{37}$$

Since $n(G_i) > n^{2/3}$, the probability for this inequality to hold for all i's between 0 and l is at least $1 - n2^{-\mu n^{4/3}}$. If this inequality holds for all i's between 0 and l, we can immediately obtain

$$l \geq \log_{1/(1-p-\epsilon)}\left(\frac{n}{n^{2/3}}\right) \tag{38}$$

$$= \frac{1}{3}\log_{1/(1-p-\epsilon)} n \tag{39}$$

$$= L(n). \tag{40}$$

I thus contains at least $L(n)$ vertices. With a probability of at least $1 - n2^{-\mu n^{4/3}}$, the above iterative procedure generates an independent set of size $L(n)$. Since $k < L(n)$, the algorithm returns 'yes' if I indeed contains $L(n)$ independent vertices, otherwise, the algorithm simply enumerates all vertex subsets in G and checks whether one of them is an independent set of size at least k. Since the procedure for generating I needs $O(n^3)$ time, the expected computation time needed for this is at most

$$O(n^3)(1 - n2^{-\mu n^{4/3}}) + 2^{O(n)}n2^{-\mu n^{4/3}} = O(n^3), \tag{41}$$

where the equality is due to the fact that the second term is bounded by a constant when n is sufficiently large. The algorithm thus needs an expected time $2^{O(k^2)} + O(n^3)$, the theorem has been proved. ■

4. Approximate algorithm

As discussed in the introduction, the maximum independent set problem cannot be approximated within a ratio of $n^{1-\epsilon}$ in polynomial time unless P = NP, where ϵ is any positive constant. In [3], it is shown that the maximum independent set in a graph can be approximated within a ratio of $O(n/\log_2^2 n)$. In [11], the approximation ratio is improved to $O(n \log_2^2 \log_2 n/\log_2^3 n)$. The result so far remains the best known approximation ratio achieved for this problem in general graphs. In [15, 18, 24], a polynomial time algorithm that can approximate the maximum independent set in a random graph within a constant ratio with high probability is developed and analysed. However, the approximation ratio of the algorithm is not guaranteed to be constant for all graphs. We show that the maximum independent set in a random graph can be approximated within a ratio of $2n/2^{\sqrt{\log_2 n}}$ in expected polynomial time, which is a significant improvement compared with the best known approximate ratio for this problem in general graphs.

THEOREM 4.1 *Given a random graph $G = (V, p)$ in n vertices where p is a positive constant between 0 and 1, the maximum independent set in G can be approximated within a ratio of $2n/2^{\sqrt{\log_2 n}}$ in expected polynomial time.*

Proof We use the following simple algorithm to compute an independent set in G. We let $k = \lfloor 2^{\sqrt{\log_2 n}} \rfloor$ and partition the vertices in G into l disjoint vertex subsets such that $l - 1$ of them contains k vertices and the remaining one contains at most k vertices. We use $G_1, G_2, \ldots, G_l$ to denote the subgraph induced by vertices in these vertex subsets. It is not difficult to see that $l \leq \lfloor n/k \rfloor + 1$.

We then use the algorithm we have developed in Theorem 2.1 to compute a maximum independent set in each of $G_1, G_2, \ldots, G_l$ and return the one that contains the largest number of vertices.

We first show that the algorithm returns an independent set in expected polynomial time. $G_1, G_2, \ldots, G_l$ are disjoint and the expected time needed to compute a maximum independent set in each of them is at most $2^{c \log_2^2 k}$, where c is some positive constant that only depends on p. Since $k \leq 2^{\sqrt{\log_2 n}}$, the expected computation time needed to compute the maximum independent set in one subgraph is at most $2^{c \log_2 n} = n^c$. The algorithm thus returns an independent set in expected time n^{c+1}.

We then show that the algorithm can achieve an approximate ratio of $2n/2^{\sqrt{\log_2 n}}$. We use APX(G) to denote the size of the independent set returned by the algorithm and OPT(G) to denote the size of a maximum independent set in G. we assume that I is a maximum independent set in G. Since we have partitioned the graph G into l disjoint subgraphs $G_0, G_1, \ldots, G_l$, at least one of the l subgraphs contains at least $\mathrm{OPT}(G)/l$ vertices from I. These vertices form an independent set in the subgraph. Since the algorithm computes a maximum independent set in each subgraph and returns the one with the largest size, we immediately obtain

$$\mathrm{APX}(G) \geq \frac{\mathrm{OPT}(G)}{l}, \tag{42}$$

this suggests that

$$\begin{aligned}
\frac{\mathrm{OPT}(G)}{\mathrm{APX}(G)} &\leq l && (43)\\
&\leq \left\lfloor \frac{n}{k} \right\rfloor + 1 && (44)\\
&\leq \frac{n}{k} + 1 && (45)\\
&\leq \frac{n}{2^{\sqrt{\log_2 n}} - 1} + 1 && (46)\\
&\leq \frac{2n}{2^{\sqrt{\log_2 n}}}. && (47)
\end{aligned}$$

The second inequality is due to the fact that $l \leq \lfloor n/k \rfloor + 1$. The fourth inequality is due to the fact that $k \geq 2^{\sqrt{\log_2 n}} - 1$. The last inequality holds for sufficiently large n. The theorem thus has been proved. ■

5. Conclusions

In this paper, we study the independent set problem in random graphs. We show that a maximum independent set in a random graph can be computed in expected subexponential time. We also show that the parameterized independent set problem is fixed parameter tractable with high probability for random graphs. Using techniques based on enumeration, we show that the largest common subgraph in two random graphs can be computed in expected subexponential time. Our work also suggests that the maximum independent set in a random graph can be approximated within a ratio of $2n/2^{\sqrt{\log_2 n}}$ in expected polynomial time, which significantly improves on the best known approximate ratio for this problem in general graphs.

It remains unknown whether the maximum independent set in a random graph can be computed in expected polynomial time or not. One possible direction of future work is to study whether there exists such an algorithm. Another related open question is that if such an algorithm does not exist, whether it can be approximated within an improved ratio in expected polynomial time. Further investigations are needed to solve these problems.

Acknowledgments

The majority of this work is done while the author was working in the Department of Mathematics and Computer Science at University of Maryland Eastern Shore, USA. This work is partially supported by the University Research Funding at Jiangsu University of Science and Technology, China, under the funding number 635301202.

References

[1] F.N. Abu-Khzam, N.F. Samatova, M.A. Rizk, and M.A. Langston, *The Maximum Common Subgraph Problem: Faster Solutions via Vertex Cover*, Proceedings of 2007 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2007), IEEE Computer Society, Washington, DC, 2007, pp. 367–373.

[2] R. Battiti and M. Protasi, *Reactive local search for the maximum clique problem*, Algorithmica 29(4) (2001), pp. 610–637.

[3] R. Boppana and M. Halldórson, *Approximating maximum independent sets by excluding subgraphs*, BIT Comput. Sci. Numer. Math. 32(2) (1994), pp. 180–196.

[4] J. Chen, X. Huang, I.A. Kanj, and G. Xia, *Linear FPT Reductions and Computational Lower Bounds*, Proceedings of the 36th ACM Symposium on Theory of Computing (STOC 2004), ACM, New York, 2004, pp. 212–221.

[5] J. Chen, X. Huang, I.A. Kanj, and G. Xia, *Strong computational lower bounds via parameterized complexity*, J. Comput. Syst. Sci. 72(8) (2006), pp. 1346–1367.

[6] I. Dinur and S. Safra, *The Importance of Being Biased*, Proceedings of the 34th ACM Symposium on Theory of Computing (STOC 2002), ACM, New York, 2002, pp. 33–42.

[7] R.G. Downey and M.R. Fellows, *Parameterized Complexity*, Springer, New York, 1998.

[8] R.G. Downey and M.R. Fellows, *Fixed parameter tractability and completeness I: basic theory*, SIAM J. Comput. 24 (1995), pp. 873–921.

[9] R.G. Downey and M.R. Fellows, *Fixed parameter tractability and completeness II: completeness for W[1]*, Theor. Comput. Sci. A 141 (1995), pp. 109–131.

[10] P. Erdős and A. Rényi, *On random graphs*, Publ. Math. 6 (1959), pp. 290–297.

[11] U. Fiege, *Approximating maximum clique by removing subgraphs*, SIAM J. Discrete Math. 18(2) (2004), pp. 219–225.

[12] U. Fiege and E. Ofek, *Finding a maximum independent set in a sparse random graph*, SIAM J. Discrete Math. 22(2) (2008), pp. 693–718.

[13] F.V. Fomin, F. Grandoni, and D. Kratsch, *Measure and Conquer: A Simple* $O(2^{0.288n})$ *Independent Set Problem*, Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA 2006), ACM, New York, 2006, pp. 18–25.

[14] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (*A Series of Books in the Mathematical Sciences*), W.H. Freeman, San Francisco, CA, 1979.

[15] G.R. Grimmett and C.J.H. Mcdiarmid, *On colouring random graphs*, Math. Proc. Camb. Philos. Soc. 77(2) (1975), pp. 313–324.

[16] A. Grosso, M. Locatelli, and F.D. Croce, *Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem*, J. Heuristics 10(2) (2004), pp. 135–152.

[17] J. Håstad, *Clique is Hard to Approximate within* $n^{1-\epsilon}$, Proceedings of the 37th Annual Symposium on Foundations of Computer Science (STOC 1996), ACM, New York, 1996, pp. 627–636.

[18] S. Homer and M. Peinado, *On the performance of polynomial-time CLIQUE approximation algorithms on very large graphs*, in *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, D.S. Johnson, ed., Amer. Mathematical Society, Providence, RI, 1993, pp. 103–124 .

[19] T. Jian, *An* $O(2^{0.308n})$ *algorithm for solving maximum independent set problem*, IEEE Trans. Comput. 35(9) (1986), pp. 847–851.

[20] D.S. Johnson, *Approximate algorithms for combinatorial problems*, J. Comput. Syst. Sci. 9 (1974), pp. 256–278.

[21] R.M. Karp, *The probability analysis of some combinatorial search problems*, in *Algorithms and Complexity: New Directions and Recent Results*, J.F. Traub, ed., Vols. 1–19, Academic Press, New York, 1976.

[22] K. Katayama, A. Hamamoto, and H. Narihisa, *An effective local search for the maximum clique problem*, Inf. Process. Lett. 95(5) (2005), pp. 503–511.

[23] J. Konc and D. Janežič, *An improved branch and bound algorithm for the maximum clique problem*, MATCH Commun. Math. Comput. Chem. 58(3) (2007), pp. 569–590.

[24] A. Coja-Oghlan and C. Efthymiou, *On Independent Sets in Random Graphs*, Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011), ACM, New York, 2011, pp. 136–144.

[25] J.M. Robson, *Finding a maximum independent set in time* $O(2^{n/4})$, Tech. Rep. 1251-01, LaBRI Université de Bordeaux I, 2001.