Colouring Random Graphs in Expected Polynomial Time

Amin Coja-Oghlan and Anusch Taraz

Humboldt-Universität zu Berlin, Institut für Informatik,
Unter den Linden 6, 10099 Berlin, Germany
{coja,taraz}@informatik.hu-berlin.de

Abstract. We investigate the problem of colouring random graphs $G \in G(n,p)$ in polynomial expected time. For the case $p < 1.01/n$, we present an algorithm that finds an optimal colouring in linear expected time. For sufficiently large values of p, we give algorithms which approximate the chromatic number within a factor of $O(\sqrt{np})$. As a by-product, we obtain an $O(\sqrt{np}/\ln(np))$-approximation algorithm for the independence number which runs in polynomial expected time provided $p \gg \ln^6 n/n$.

1 Introduction and Results

The problem of determining the minimum number of colours needed to colour a graph G – denoted by the $\chi(G)$, the chromatic number of G – was proven to be NP-hard by Karp [19]. In fact, already deciding whether a given graph is 3-colourable is NP-complete. These results have been completed in the last decade by non–approximability theorems. Feige and Kilian [9] showed that, unless $coRP = NP$, no polynomial time algorithm with approximation ratio less than $n^{1-\varepsilon}$ exists.

However, these hardness results are deeply rooted in the worst–case paradigm and thus the question arises whether algorithms can be designed and analyzed that perform well on *average* or *random* instances. The binomial model for such a *random graph*, usually denoted by $G(n,p)$, is defined as follows. In a graph with vertex set $[n] = \{1,\ldots,n\}$ every possible edge is present with probability p independently of all others. Here $p = p(n)$ can be a function in n. In the particular case of $G(n,\frac{1}{2})$ this is exactly the uniform distribution on the set of all graphs with vertex set $[n]$. Admittedly, the $G(n,p)$-model may not be appropriate in every setting, but it is definitely the standard model of a random graph. We shall only mention those results on random graphs here which are important for our purposes and refer the interested reader to [4, 16, 13] for general background information and to the survey [24] for a comprehensive overview on algorithmic random graph colouring.

We say that $G(n,p)$ has a certain property A *almost surely* or *with high probability*, if the probability that $G \in G(n,p)$ has A tends to 1 as $n \to \infty$. Let $b = 1/(1-p)$. Bollobás and Łuczak (cf. [4, 16]) proved that almost surely a

random graph $G \in G(n,p)$ satisfies

$$\chi(G) \sim \begin{cases} \frac{n}{2\log_b n} & \text{for constant } p \\ \frac{np}{2\ln np} & \text{for } \frac{C}{n} < p = o(1) \end{cases} \qquad \alpha(G) \sim \begin{cases} 2\log_b n & \text{for constant } p \\ \frac{2\ln np}{p} & \text{for } \frac{C}{n} < p = o(1), \end{cases} \quad (1)$$

where C is a sufficiently large constant. For even smaller values of p, and constant k, the situation is as follows. For any number $k \geq 2$, there are constants c_k^- and c_k^+ such that for $p \leq (1-\varepsilon)c_k^-/n$ (and $p \geq (1+\varepsilon)c_k^+/n$ respectively), $G(n,p)$ almost surely is (respectively is not) k-colourable. In fact, it is conjectured that one can actually choose $c_k^- = c_k^+$ [1].

These structural issues are accompanied by the obvious algorithmic question: are there good algorithms to compute a (near-) optimal colouring of $G \in G(n,p)$? The greedy colouring algorithm achieves good results in this setting. The algorithm considers the vertices in an arbitrary order and assigns to each vertex the smallest possible colour. Grimmett and McDiarmid [14] proved that almost surely the number of colours used for $G \in G(n, \frac{1}{2})$ is asymptotic to $n/\log_2 n$. A slight improvement over the greedy algorithm using randomization has been obtained by Krivelevich and Sudakov [25]. Still, to date the best known approximation ratio is asymptotically 2.

It is worth emphasizing that for inputs from $G(n,p)$ the greedy algorithm *always* has polynomial running time (in fact, linear) and achieves a 2-approximation *with high probability.* Nevertheless, the algorithm itself does not find a certificate for a lower bound on the chromatic number of the input graph G, and thus it cannot guarantee a good approximation ratio. Indeed, there are input instances for which the approximation ratio achieved by the greedy algorithm is quite bad (i.e. close to n), even for almost all permutations of the vertices [26]. This motivates the following question by Karp [20]:

> Is there an algorithm that for inputs from $G(n,p)$ has *expected* polynomial running time and *always* uses the minimum number of colours? (2)

Here the expected running time of an algorithm $\mathcal{A}$ is defined as $\sum_G R_{\mathcal{A}}(G)\mathrm{P}(G)$, where the sum runs over all graph G with vertex set $[n]$, $R_{\mathcal{A}}(G)$ is the running time needed by $\mathcal{A}$ for input G, and $\mathrm{P}(G)$ denotes the probability that G is chosen according to the distribution $G(n,p)$.

The obvious approach to design an algorithm that meets the requirements of (2) is as follows. Typical inputs will have certain structural properties which enable us to efficiently find an optimal solution. In the exceptional cases, we have to invest more time, but this doesn't affect the expected running time much as it will happen only with small probability. However, the crucial point of this approach is *how to decide efficiently whether the input is typical or not.*

Our first result answers question (2) in the affirmative for the case of small edge probabilities p.

Theorem 1. *For $p \leq 1.01/n$ there exists a colouring algorithm which finds an optimal colouring and, when applied to $G(n,p)$, has linear expected running time.*

We remark that the constant 1.01 is certainly not best possible. It merely demonstrates that the algorithm can find optimal colourings even after $G(n,p)$ has passed the *phase transition* where it suddenly becomes much more complex. Variants of question (2) have been addressed by several authors (see e.g. [5, 7, 12, 29]). They, too, give exact algorithms with expected polynomial running time, but over a different probability distribution than $G(n,p)$.

A new line of research has recently been initiated by Krivelevich and Vu [22], relaxing question (2) in the following way. Is there an algorithm, that for inputs from $G(n,p)$ has *expected* polynomial running time and *always* approximation ratio r? Here r may be a constant or a function in n (and p).

Krivelevich and Vu prove in [22] that for $n^{-1/2+\varepsilon} \leq p \leq 0.99$, there exists a colouring algorithm with approximation ratio $O(\sqrt{np}/\ln n)$ and expected polynomial running time over $G(n,p)$. Moreover they ask [22, 24] whether it is possible to find similarly good approximation algorithms for smaller values of p. Our next two results give positive answers to this question.

Theorem 2. *For $\ln(n)^6/n \ll p \leq 3/4$ there exists a colouring algorithm with approximation ratio $O(\sqrt{np})$ and polynomial expected running time over $G(n,p)$.*

For smaller values of p, where the above theorem does not hold, we can prove the following.

Theorem 3. *Let $p \geq 1/n$. Then there exists a colouring algorithm with approximation ratio $\frac{e^2}{3}np$ and linear expected running time over $G(n,p)$.*

Observe that in the range where p is too small for Theorem 2 to apply, Theorem 3 is almost as good: here both approximation ratios are of order $poly(\ln n)$.

Instead of considering algorithms that find (near-) optimal colourings for a given graph, one also considers the problem of deciding, for some integer k, whether the input graph is k-colourable or not. Krivelevich [23] has shown that for every fixed $k \geq 3$ there exists a constant $C = C(k)$ so that k-colourability can be decided in expected polynomial time over $G(n,p)$, provided that $p(n) \geq C/n$. Furthermore, he asks [24] whether it is possible to find such algorithms (i) for the case where $k = k(n)$ is a function growing in n and (ii) for any value of the edge probability $p = p(n)$. It turns out that the methods we have developed for our colouring algorithms contribute to answering the above questions.

Our techniques also capture a more general *semirandom* situation. Semirandom graph problems have been studied e.g. in [10, 5]. In contrast to $G(n,p)$, semirandom settings allow for an adversary to change the outcome of the random experiment to a certain degree. In this way they capture a larger class of input distributions and therefore require rather robust algorithmic techniques. Here we introduce two semirandom models for the decision version of k-colouring, $G(n,p)^+$ and $G(n,p)^-$, and give algorithms for deciding k-colourability. In the model $G(n,p)^+$, instances are created in two steps as follows:

1. First, a random graph $G_0 = G(n,p)$ is chosen.
2. Then, an adversary may *add* edges to G_0, in order to produce the input instance G.

Note that the adversary can change the vertex degrees, the independence number, the chromatic number and even the spectrum of the random graph G_0. Instances of the semirandom model $G(n,p)^-$ are created similarly, but the adversary is only allowed to *remove* edges instead of adding edges. Given $G_0 = G(n,p)$, let $\mathcal{I}(G_0)^+$ denote the set of all *instances over* G_0, i.e. the set of all graphs G that can be produced by adding edges to G_0. We define $\mathcal{I}(G_0)^-$ similarly. An algorithm $\mathcal{A}$ is said to run in *expected polynomial time applied to the semirandom model* $G(n,p)^+$ if there is a constant l such that for any map I that to each graph G_0 associates an element $I(G_0) \in \mathcal{I}(G_0)^+$, we have

$$\sum_{G_0} R_{\mathcal{A}}(I(G_0))P(G_0) = O(n^l),$$

where $R_{\mathcal{A}}(I(G_0))$ denotes the running time of $\mathcal{A}$ on input $I(G_0)$. We define a similar notion for the model $G(n,p)^-$.

Theorem 4. *For any $k = k(n)$ and $p \leq k/(10n)$ there exists an algorithm which decides whether a graph G is k-colourable, and, when applied to $G(n,p)^-$, has linear expected running time.*

Theorem 5. *Let $k = k(n)$ and $p = p(n) \geq \max\{\ln(n)^7/n, 200k^2/n\}$. There exists an algorithm which decides whether a graph G is k-colourable, and, when applied to $G(n,p)^+$, has polynomial expected running time.*

Observe that Theorem 4 deals with the range where $G(n,p)$ is almost surely k-colourable, while Theorem 5 concerns graphs which almost surely are not k–colourable. The threshold for k-colourability has order $k \ln k/n$, which lies in the remaining gap for $\Omega(k/n) \leq p \leq \tilde{O}(k^2/n)$.

As an immediate by-product from our work on graph colouring, we obtain the following approximation algorithm for the independent set problem.

Theorem 6. *For $\ln(n)^6/n \ll p \leq 3/4$ there exists an algorithm that approximates the independence number with approximation ratio $O(\sqrt{np}/\ln(np))$ and has polynomial expected running time over $G(n,p)$.*

The remainder of the paper is organized as follows. After introducing the necessary technical terminology, we shall first prove Theorem 3 in Section 2, as this is the simplest algorithm. Section 3 deals with Theorem 1. The algorithms for Theorems 2 and 6 can be found in Section 4. Finally, in Section 5 we give the algorithms for Theorems 4 and 5.

The Euler constant will be denoted by $\mathrm{e} = \exp(1)$. For a graph $G = (V,E)$ and a subset $S \subseteq V$ we let $G[S]$ be the subgraph of G induced by S. We write $H \subseteq G$ if H is a (weak) subgraph of G. Denote by $|G| = |V|$ the order of G. For a vertex v in G we let $d(v) = d_G(v)$ be the degree of v in G. Denote by $\delta(G)$ the minimal degree in G. In this paper we disregard rounding issues since these do not affect our arguments.

2 A simple $O(np)$-approximation algorithm

Consider an integer $k \geq 2$. The *k-core* of a graph G is the unique subgraph $H \subseteq G$ of maximum cardinality such that $\delta(H) \geq k$. The following observation is elementary but important for our algorithms: if G has no k-core, then a k-colouring of G can be found in linear time. Indeed, consecutively push vertices of degree at most $k-1$ from the graph onto a stack until the graph is empty. Then, while putting the vertices back into the graph in reverse order, assign to each vertex the smallest colour possible, which is at most k since the vertex is (at that point) connected to at most $k-1$ other vertices. This procedure is known as the smallest–last heuristic [2]. Observe that the order in which we remove vertices from the graph does not matter, as long as they have degree at most $k-1$ at the time of removal.

Pittel, Spencer and Wormald [28] computed the constants γ_k which determine the precise threshold $p = \gamma_k/n$ for the appearance of the k-core in $G(n,p)$. To give an example, $\gamma_3 \approx 3.35$. By the preceding discussion, it is clear that this marks a lower bound for the threshold of k-colourability. It was an open question of Bollobás whether the appearance of the k-core in $G(n,p)$ coincides with the threshold of k-colourability. A few years ago, Achlioptas and Molloy [1] answered this in the negative by showing that $G(n,p)$ remains 3-colourable for at least as long as $p \leq 3.84/n$.

Let $k \geq 3$. We emphasize that even though almost surely $G(n,p)$ does not contain a k-core of size, say, $\ln^{10}(n)$ at $p = 1/n$, the probability for this event is not exponentially small in n, and hence such an event is not sufficiently unlikely to allow us to find the optimal colouring by brute force *on the whole graph*. Instead, the following simple algorithm will only need to deal with the core.

Algorithm 7. `CoreColour`(G,k)
Input: A graph $G = (V,E)$ and an integer k.
Output: A colouring of G.

1. Let $G' := G$, set stack $S := \emptyset$.
2. While in G' there exists a vertex v with $d_{G'}(v) \leq k-1$ do
   push v onto S, $G' := G' - v$.
3. Colour G' exactly by Lawler's algorithm [27] in time $O((1+\sqrt[3]{3})^{n'})$, where $n' = |G'|$.
4. Extend this to a colouring of G: while $S \neq \emptyset$ do
   pick the top vertex v from S, assign to v the least possible colour.

Proposition 8. *For any graph G and any integer $k \geq 2$, `CoreColour`(G,k) finds a colouring of G with at most $\max\{\chi(G), k\}$ colours.*

Proof. Step 3 uses $\chi(G') \leq \chi(G)$ colours. Let v be an arbitrary vertex. Since $d_{G'}(v) \leq k-1$ when v was removed, Step 4 will never need a colour greater than k, since v has at most $k-1$ coloured neighbours at the moment it is coloured. □

Observe next that at the beginning of step 3, the vertices in G' make up the k-core. Therefore, in order to examine the expected running time of `CoreColour`,

we need an upper bound on the probability that G contains a k-core of a certain size.

Lemma 9. *Let $p \geq 1/n$ and consider an integer $k \geq \mathrm{e}^2 np$. Then*

$$P(G(n,p) \text{ contains a } k\text{-core on } \nu \text{ vertices}) < \mathrm{e}^{-\nu}$$

Proof. The probability that there exists a k-core on ν vertices is bounded from above by the probability that there exists a weak subgraph H on ν vertices with $\mu = k\nu/2$ edges. Let $N = \binom{\nu}{2}$. Then the expected number of such subgraphs is given by

$$\binom{n}{\nu}\binom{N}{\mu}p^\mu \leq \left(\frac{\mathrm{e}n}{\nu}\right)^\nu \left(\frac{\mathrm{e}\nu^2 p}{2\mu}\right)^\mu \leq \left[\frac{\mathrm{e}n}{\nu}\left(\frac{\mathrm{e}\nu p}{k}\right)^{k/2}\right]^\nu \leq \left[\left(\frac{\nu}{\mathrm{e}n}\right)^{\frac{k}{2}-1}\right]^\nu \leq \mathrm{e}^{-\nu},$$

where the last step used $\nu \leq n$ and $k \geq \mathrm{e}^2 np \geq 4$ because of $p \geq 1/n$. □

Proof of Theorem 3. We can easily check whether $\chi(G) \leq 2$, and, if so, output an optimal colouring. So assume that $\chi(G) > 2$. Set $k := \mathrm{e}^2 np$ and apply `CoreColour`$(G(n,p), k)$.

Proposition 8 shows that the algorithm has approximation ratio at most $k/3 = (\mathrm{e}^2/3)np$ as claimed. Its running time is linear for step 2 and step 4. As $1 + \sqrt[3]{3} < 2.443 < \mathrm{e}$, Lemma 9 implies that the expected running time of step 3 can be bounded by $\sum_{\nu=0}^{n}(1+\sqrt[3]{3})^\nu \cdot \mathrm{P}(|G'| = \nu) = O(n)$. □

3 Finding an optimal colouring

This section contains a sketch of the proof of Theorem 1. The underlying idea is basically the same as in Section 2, but we need to refine both the algorithm and the computations in order to achieve an approximation ratio 1.

We will employ an exact algorithm by Beigel and Eppstein [3] which decides in time $O(1.3447^n)$ whether a given graph is 3-colourable or not. Suppose that it answers this question in the affirmative, then it is obviously not hard to actually find such a 3-colouring in $O(n^2 1.3447^n)$ steps, simply by trying to add edges to the graph as long as the graph remains 3-colourable. Thus in the end we have a complete 3-partite graph and can read off the colouring.

The basic idea of our algorithm is as follows: having checked for 2-colourability, we peal off vertices until we get to the 3-core. For the 3-core we merely decide whether it is 3-colourable because applying the Beigel–Eppstein *decision* algorithm is much cheaper than *finding* an optimal colouring via Lawler's algorithm. If yes, we are done. If not, then we know that $\chi(G) \geq 4$ and continue shaving off vertices until we get to the 4-core, which will most probably be substantially smaller than the 3-core. Now we optimally colour the 4-core using Lawler's algorithm.

Algorithm 10. `ExactColour`(G)
Input: A graph $G = (V, E)$
Output: A colouring of G.

1. If $\chi(G) \leq 2$ then find an optimal colouring of G and stop.
2. Let $G' := G$, set stack $S := \emptyset$.
   While there exists a vertex v with $d_{G'}(v) \leq 2$ do
   push v onto S, let $G' := G' - v$.
3. Check whether $\chi(G') \leq 3$ using Beigel and Eppstein's algorithm.
4. If yes then find a 3-colouring of G' in time $O(n'^2 1.3447^{n'})$, where $n' = |G'|$.
5. If no then run `CoreColour`$(G', 4)$.
6. Extend the obtained colouring of G' to a colouring of G as follows:
   while $S \neq \emptyset$ do
   pick the top vertex v from S, assign to v the least possible colour.

By similar arguments as in Section 2, it is clear that the algorithm produces an optimal colouring.

In order to show that the expected running time remains indeed linear for p as large as $1.01/n$, one needs to sharpen the bound given in Lemma 9. Apart from obvious room for improvements by using more careful technical estimates, most of the gain is won by only counting those subgraphs which have indeed minimum degree k and not just $k\nu/2$ edges.

4 $O(\sqrt{np})$-approximation via the ϑ-function

Throughout this section we shall assume that $p \gg \ln(n)^6/n$. Let $\omega = np$ and $G \in G(n,p)$ be a random graph. On input $(G, 10np)$, the algorithm `CoreColour` presented in Section 2 produces a colouring of G using at most $\max\{\chi(G), 10np\}$ colours. Thus, `CoreColour` immediately gives an $O(np)$-approximation algorithm for colouring $G(n,p)$ within expected polynomial time. In order to improve the approximation ratio $O(np)$, it is crucial to develop techniques to compute good lower bounds on the chromatic number of the input graph, since we need to distinguish graphs with a chromatic number near the "typical value" (1) from "pathological cases" efficiently. To this end, we shall use the following trivial inequality. For any graph G of order n we have $\chi(G) \geq n/\alpha(G)$. Thus, instead of lower bounding the chromatic number $\chi(G)$, we may also compute an upper bound on the independence number $\alpha(G)$.

The same approach has been used in [22], where the independence number of $G(n,p)$ is bounded from above as follows. Given a random graph $G \in G(n,p)$ with vertex set $V = \{1, \ldots, n\}$, let the matrix $M = (m_{ij})_{i,j \in V}$ have entries

$$m_{ij} = \begin{cases} 1 & \text{if } \{i,j\} \notin E(G) \text{ or } i = j \\ \frac{p-1}{p} & \text{otherwise.} \end{cases}$$

Then the largest eigenvalue $\lambda_1(M)$ provides an upper bound on $\alpha(G)$. In order to base an approximation algorithm with polynomial expected running time on

computing the bound $n/\lambda_1(M) \leq \chi(G)$, it is necessary to estimate the probability that $\lambda_1(M)$ is "large". To this end, it is shown in [22] that

$$P(\lambda_1(M) \geq 4(n/p)^{1/2}) \leq 2^{-\omega/8}. \tag{3}$$

Note that the exponent on the right hand side depends on ω (and thus both on p and n). Nonetheless, in the case $\omega \geq n^{1/2+\varepsilon}$, $\varepsilon > 0$, the concentration result (3) suffices to obtain an $O(\sqrt{np}/\ln n)$-approximation algorithm for both the chromatic number and the independence number of $G(n,p)$. The proof of (3) is based on Talagrand's inequality (cf. [16]).

However, in the case $p \ll n^{-1/2}$, (3) seems not to be sharp enough in order to construct an algorithm with expected polynomial running time and approximation ratio $\tilde{O}(np)^{1/2}$. Therefore, we shall use the *Lovász number* $\vartheta(G)$ as an upper bound on $\alpha(G)$ instead of the eigenvalue $\lambda_1(M)$; it is well-known that $\alpha(G) \leq \vartheta(G)$ for any graph G. For a thorough introduction to the Lovász number the reader is referred to [15, 21]. The Lovász number $\vartheta(G)$ can be computed within polynomial time, using the ellipsoid algorithm [15].

What is the relation between $\vartheta(G)$ and the eigenvalue $\lambda_1(M)$? Let us call a real symmetric matrix $A = (a_{vw})_{v,w \in V}$ *feasible for* G if $a_{vv} = 1$ for all v and $a_{uv} = 1$ for any pair u, v of non-adjacent vertices of G. Then we have $\vartheta(G) = \min\{\lambda_1(A) |\ A \text{ is feasible for } G\}$, where as above $\lambda_1(A)$ denotes the maximum eigenvalue of A. Observing that the matrix M defined above is feasible, we obtain

$$\alpha(G) \leq \vartheta(G) \leq \lambda_1(M). \tag{4}$$

Thus, our colouring algorithm uses $n/\vartheta(G)$ as a lower bound on $\chi(G)$. Consequently, we need to estimate the expectation of $\vartheta(G(n,p))$.

Lemma 11. *With high probability,* $\vartheta(G(n,p)) \leq 3(n/p)^{1/2}$.

The proof of Lemma 11 is based on (4) and on the fact that M is a random symmetric matrix as considered in [11], except for the fact that the entries m_{ij} may be unbounded. The analysis of our colouring algorithm makes use of the following concentration result on the Lovász number of random graphs [6].

Lemma 12. *Let μ be a median of $\vartheta(G(n,p))$. Let $\xi \geq \max\{10, \sqrt{\mu}\}$. Then*

$$P(\vartheta(G(n,p)) \geq \mu + \xi) \leq 30 \exp\left(-\frac{\xi^2}{5\mu + 10\xi}\right).$$

Consequently, if $a \geq 10\sqrt{n/p}$, then

$$P(\vartheta(G(n,p)) \geq a) \leq \exp(-a/30). \tag{5}$$

The proof of Lemma 12 is based on Talagrand's inequality. Note that, contrary to the estimate (3), the exponent on the right hand side of (5) does not increase as p decreases. The following algorithm for colouring $G(n,p)$ resembles the colouring algorithm proposed in [22]. The main difference is that we use the algorithm `CoreColour` instead of the greedy algorithm in step 1, and that we bound the chromatic number via the Lovász number.

Algorithm 13. `ApproxColour`(G)
Input: A graph $G = (V, E)$.
Output: A colouring of G.

1. Run `CoreColour`$(G, 10np)$. Let $\mathcal{C}$ be the resulting colouring of G.
2. Compute $\vartheta(G)$. If $\vartheta(G) \leq 10(n/p)^{1/2}$, then output $\mathcal{C}$ and terminate.
3. Check whether there exists a subset S of V, $|S| = 25\ln(np)/p$, such that $|V \setminus (S \cup N(S))| > 10(n/p)^{1/2}$, by enumerating all subsets of V of cardinality $25\ln(np)/p$. If no such set exists, then output $\mathcal{C}$ and terminate.
4. Check whether in G there is an independent set of size $10(n/p)^{1/2}$ by enumerating all subsets of V of cardinality $10(n/p)^{1/2}$. If this is not the case, then output $\mathcal{C}$ and terminate.
5. Colour G exactly by Lawler's algorithm [27] in time $O(n^3(1 + \sqrt[3]{3})^n)$.

It is straightforward to verify that `ApproxColour` achieves the approximation ratio stated in Theorem 2. The proof that the expected running time over $G(n, p)$ is polynomial is based on (5).

In [17] it is observed that with high probability $\vartheta(G(n, p)) = \Omega(n/p)^{1/2}$. Since the chromatic number $\chi(G(n, p))$ satisfies (1) with high probability, we cannot hope for an approximation ratio considerably better than $O(np)^{1/2}$ using $n/\vartheta(G(n, p))$ as a lower bound on $\chi(G(n, p))$.

Let us conclude this section noting that the techniques behind `ApproxColour` immediately lead to an approximation algorithm for the maximum independent set problem in $G(n, p)$. In order to obtain an algorithm `ApproxMIS` that satisfies the requirements of Theorem 6, we adapt `ApproxColour` as follows. Instead of `CoreColour`, we use the greedy (colouring) algorithm and keep the largest colour class it produces. Then we estimate the independence number of the input graph as `ApproxColour` does. Finally, instead of using an exact graph colouring algorithm in step 5, we use an exact algorithm for the maximum independent set problem.

Lemma 14. *The probability that the largest colour class produced by the greedy colouring algorithm is of size* $< \ln(np)/(2p)$ *is at most* 2^{-n}.

The proof uses a similar argument as given already in [22] for the case that $p \geq n^{\varepsilon - 1/2}$.

Algorithm 15. `ApproxMIS`(G)
Input: A graph $G = (V, E)$.
Output: An independent set of G.

1. Run the greedy algorithm for graph colouring on input G. Let I be the largest resulting colour class. If $|I| < \ln(np)/(2p)$, then go to 5.
2. Compute $\vartheta(G)$. If $\vartheta(G) \leq 10(n/p)^{1/2}$, then output I and terminate.
3. Check whether there exists a subset S of V, $|S| = \frac{25\ln(np)}{p}$, such that $|V \setminus (S \cup N(S))| > 10(n/p)^{1/2}$. If no such set exists, then output I and terminate.
4. Check whether in G there is an independent set of size $10\sqrt{n/p}$. If this is not the case, then output I and terminate.
5. Enumerate all subsets of V and output a maximum independent set.

5 Deciding k-colourability

In this final section we shall apply the methods established so far to the following problem. Given a semirandom graph $G = G(n,p)^+$ or $G = G(n,p)^-$, we are to decide within polynomial expected time whether G is $k = k(n)$-colourable. First, let us see what the algorithm `CoreColour` contributes to the problem of deciding k-colourability in the model $G(n,p)^-$.

Algorithm 16. `CoreDecide`(G)
Input: A graph $G = (V,E)$.

1. Run `CoreColour`$(G, 10np)$. Let $\mathcal{C}$ be the resulting colouring of G.
2. If the number of colours used in $\mathcal{C}$ is at most k, output "G is k-colourable". Otherwise, output "G is not k-colourable".

Let $G_0 = G(n,p)$ be a random graph, $p \le k/(10n)$, and let $G \in \mathcal{I}(G_0)^-$ be the instance chosen by the adversary. Because the adversary can only remove edges, the $10np$-core of G is contained in the $10np$-core of G_0. Thus, the argument given in the proof of Lemma 9 shows that the expected running time of `CoreDecide` is linear. Moreover, it is easy to see that the answer produced by `CoreDecide` is always correct. For if `CoreDecide` states that G is k-colourable, then `CoreColour` must have found a proper k-colouring of G. Conversely, by Proposition 8 we know that the number of colours that `CoreColour` uses is at most $\max\{\chi(G), 10np\}$. Thus, by our choice of p, if $\chi(G) \le k$, then `CoreColour` finds a k-colouring of G.

Let us now assume that $np \ge \max\{200k^2, \ln(n)^7\}$. In this range we almost surely have $\chi(G(n,p)) > k$. Applying the techniques behind `ApproxColour` to the problem of deciding k-colourability of the semirandom graph $G(n,p)^+$, we obtain the following algorithm.

Algorithm 17. `Decide`(G)
Input: A graph $G = (V,E)$.

1. Compute $\vartheta(G)$. If $\vartheta(G) \le 10(n/p)^{1/2}$, then terminate with output "G is not k-colourable".
2. Check whether there exists a subset S of V, $|S| = 25\ln(np)/p$, such that $|V \setminus (S \cup N(S))| > 10(n/p)^{1/2}$. If no such set exists, then output "G is not k-colourable" and terminate.
3. Check whether in G there is an independent set of size $10(n/p)^{1/2}$. If this is not the case, then output "G is not k-colourable" and terminate.
4. Compute the chromatic number of G exactly by Lawler's algorithm [27] in time $O(3^n)$ and answer correctly.

In order to prove that `Decide` has polynomial expected running time applied to $G(n,p)^+$, we consider a random graph $G_0 \in G(n,p)$ and an instance $G \in \mathcal{I}(G_0)^+$. Because G_0 is a subgraph of G, we have $\vartheta(G) \le \vartheta(G_0)$, by the monotonicity of the ϑ-function (cf. [21]). Moreover, if G admits a set S as in step 2 of `Decide`, then a fortiori G_0 admits such a set S. Finally, $\alpha(G) \le \alpha(G_0)$.

Thus, the arguments that we used to prove that the expected running time of `ApproxColour` applied to $G(n,p)$ is polynomial immediately yield that the expected running time of `Decide` applied to the semirandom model $G(n,p)^+$ is polynomial.

We finally claim that `Decide`(G) always answers correctly. For assume that `Decide`(G) asserts that G is k-colourable. Then step 4 of `Decide`(G) must have found a proper k-colouring of G; hence, G is k-colourable. On the other hand, assume that `Decide`(G) asserts that G is not k-colourable. Then, `Decide`(G) has found out that

$$\alpha(G) \leq 10(n/p)^{1/2} + 25\ln(np)/p \leq 11(n/p)^{1/2}.$$

Thus,

$$\chi(G) \geq \frac{n}{\alpha(G)} \geq \frac{n}{11(n/p)^{1/2}} = \frac{(np)^{1/2}}{11} > k,$$

as claimed.

6 Conclusion

No serious attempts have been made to optimize the constants involved. For example, an easy way to slightly enlarge the range of p for which the algorithm `ExactColour` has expected polynomial running time would be to use Eppstein's recent algorithm [8] for finding an optimal colouring instead of Lawler's – with $O(2.415^n)$ running time instead of $O(2.443^n)$. However it is of course clear that our general approach must fail at the latest when $p = 3.84/n$, as here the 3-core appears almost surely and has linear size.

References

1. Achlioptas, D., Molloy, M.: The analysis of a list-coloring algorithm on a random graph, Proc. 38th. IEEE Symp. Found. of Comp. Sci. (1997) 204-212
2. Beck, L.L., Matula, D.W.: Smallest-last ordering and clustering and graph coloring algorithms, J. ACM **30** (1983) 417–427
3. Beigel, R., Eppstein, D.: 3-coloring in time $O(1.3446^n)$: a no-MIS algorithm, Proc. 36th. IEEE Symp. Found. of Comp. Sci. (1995) 444-453
4. Bollobás, B.: Random graphs, 2nd edition, Cambridge University Press 2001
5. Coja-Oghlan, A.: Finding sparse induced subgraphs of semirandom graphs. Proc. 6th. Int. Workshop Randomization and Approximation Techniques in Comp. Sci. (2002) 139–148
6. Coja-Oghlan, A.: Finding large independent sets in expected polynomial time. To appear in Proc. STACS 2003
7. Dyer, M., Frieze, A.: The solution of some NP-hard problems in polynomial expected time, J. Algorithms **10** (1989) 451–489
8. Eppstein, D.: Small maximal independent sets and faster exact graph coloring. To appear in J. Graph Algorithms and Applications

9. Feige, U., Kilian, J.: Zero knowledge and the chromatic number. Proc. 11. IEEE Conf. Comput. Complexity (1996) 278–287
10. Feige, U., Kilian, J.: Heuristics for semirandom graph problems. J. Comput. and System Sci. **63** (2001) 639–671
11. Füredi, Z., Komloś, J.: The eigenvalues of random symmetric matrices, Combinatorica **1** (1981) 233–241
12. Fürer, M., Subramanian, C.R., Veni Madhavan, C.E.: Coloring random graphs in polynomial expected time. Algorithms and Comput. (Hong Kong 1993), Springer LNCS 762, 31–37
13. Frieze, A., McDiarmid, C.: Algorithmic theory of random graphs. Random Structures and Algorithms **10** (1997) 5–42
14. Grimmett, G., McDiarmid, C.: On colouring random graphs. Math. Proc. Cam. Phil. Soc **77** (1975) 313–324
15. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization. Springer 1988
16. Janson, S., Łuczak, T., Ruciński, A.: Random Graphs. Wiley 2000
17. Juhász, F.: The asymptotic behaviour of Lovász ϑ function for random graphs, Combinatorica **2** (1982) 269–280
18. Karger, D., Motwani, R., Sudan, M.: Approximate graph coloring by semidefinite programming. Proc. of the 35th. IEEE Symp. on Foundations of Computer Science (1994) 2–13
19. Karp, R.: Reducibility among combinatorial problems. In: Complexity of computer computations. Plenum Press (1972) 85–103.
20. Karp, R.: The probabilistic analysis of combinatorial optimization algorithms. Proc. Int. Congress of Mathematicians (1984) 1601–1609.
21. Knuth, D.: The sandwich theorem, Electron. J. Combin. **1** (1994)
22. Krivelevich, M., Vu, V.H.: Approximating the independence number and the chromatic number in expected polynomial time. J. of Combinatorial Optimization **6** (2002) 143–155
23. Krivelevich, M.: Deciding k-colorability in expected polynomial time, Information Processing Letters **81** (2002) 1–6
24. Krivelevich, M.: Coloring random graphs – an algorithmic perspective, Proc. 2nd Coll. on Mathematics and Computer Science, B. Chauvin et al. Eds., Birkhauser, Basel (2002) 175-195.
25. Krivelevich, M., Sudakov, B.: Coloring random graphs. Informat. Proc. Letters **67** (1998) 71–74
26. Kučera, L.: The greedy coloring is a bad probabilistic algorithm. J. Algorithms **12** (1991) 674–684
27. Lawler, E.L.: A note on the complexity of the chromatic number problem, Information Processing Letters **5** (1976) 66–67
28. Pittel, B., Spencer, J., Wormald, N.: Sudden emergence of a giant k-core in a random graph. JCTB **67** (1996) 111–151
29. Prömel, H.J., Steger, A.: Coloring clique-free graphs in polynomial expected time, Random Str. Alg. **3** (1992) 275–302