Combinatorics, Probability and Computing (2000) 9, 45–77. Printed in the United Kingdom

Algorithms for Colouring Random *k*-colourable Graphs

C. R. SUBRAMANIAN[†]

Max-Planck Institut für Informatik,
Im Stadtwald, 66123, Saarbrücken, Germany.
(e-mail: crs@mpi-sb.mpg.de)

Received 24 March 1998; revised 25 October 1998

The k-colouring problem is to colour a given k-colourable graph with k colours. This problem is known to be NP-hard even for fixed $k \geqslant 3$. The best known polynomial time approximation algorithms require n^{δ} (for a positive constant δ depending on k) colours to colour an arbitrary k-colourable n-vertex graph. The situation is entirely different if we look at the average performance of an algorithm rather than its worst-case performance. It is well known that a k-colourable graph drawn from certain classes of distributions can be k-coloured almost surely in polynomial time.

In this paper, we present further results in this direction. We consider k-colourable graphs drawn from the random model in which each allowed edge is chosen independently with probability $p(n)$ after initially partitioning the vertex set into k colour classes. We present polynomial time algorithms of two different types. The first type of algorithm always runs in polynomial time and succeeds almost surely. Algorithms of this type have been proposed before, but our algorithms have provably exponentially small failure probabilities. The second type of algorithm always succeeds and has polynomial running time on average. Such algorithms are more useful and more difficult to obtain than the first type of algorithms. Our algorithms work as long as $p(n) \geqslant n^{-1+\epsilon}$ where ϵ is a constant greater than $1/4$.

1. Introduction

The graph colouring problem is to determine an assignment of colours to vertices in the graph so that no two adjacent vertices receive the same colour. G is said to be *k-colourable* if there exists a k-colouring for G, that is, G has a colouring using at most k colours. The *chromatic number* of G, denoted $\chi(G)$, is the smallest value of k such that G is k-colourable. It is an important problem in algorithmic graph theory, and was shown by Karp (see [6]) to be NP-complete.

[†] This research was done while the author was a graduate student at the department of Computer Science and Automation, Indian Institute of Science, Bangalore, India.

There are at least two ways of measuring the 'running time' of a colouring algorithm. In the *worst-case complexity* measure, the algorithm is judged by its maximum running time over all relevant n-vertex input instances. In the *average-case complexity* measure, the algorithm is judged by its *average* running time with respect to a chosen probability distribution over the set of all relevant n-vertex input instances. We note that the average case performance is very sensitive to the choice of the probability distribution.

Two types of algorithms are considered within the average case model (also called random models), as follows.

Almost surely succeeding (a.s.) algorithms. These are polynomial time (worst-case) algorithms which, given a random instance, succeed on it with probability $1 - o(1)$.

Polynomial average time (p.av.t.) algorithms. These are algorithms which succeed on *all* relevant input instances and whose average running time is polynomial.

Algorithms of the first type are much easier to obtain than algorithms of the second type, because a few exponentially slow instances can spoil the whole average. Also, from a p.av.t. algorithm, we can obtain a.s. algorithms of required failure probabilities. This is explained further at the end of this section.

If we adopt the worst-case measure, $\chi(G)$ cannot be approximated within $O(n^{1-\epsilon})$ for any fixed $\epsilon > 0$, unless NP= ZPP [4]. Also, the existing approximation algorithms are only guaranteed to use $O(n^{1-3/(k+1)}(\log n)^{O(1)})$ colours even if the input is a k-colourable (fixed k) graph [9]. For more details on the worst-case complexity of graph colouring problem, we refer the reader to [16].

On the other hand, if we adopt the average case measure, we can obtain much better approximations (or even optimum solutions) in polynomial time (or in p.av.t.) for 'most' of the input graphs. For example, Grimmett and McDiarmid [7] show that the greedy algorithm for colouring uses at most twice the minimum number ($\chi(G)$) of colours on most n-vertex graphs. Turner and others [18, 3, 2] give polynomial time algorithms that colour *most* of the k-colourable graphs with k colours, for any constant k. One reason, pointed out by Karp [10] and Frieze and McDiarmid [5], is that several intractability proofs are based on very special classes of graphs which we may encounter in practice only rarely. Thus, 'average case analysis banishes the pessimism of worst-case analysis', in the words of Frieze and McDiarmid [5].

1.1. Description of average-case models

Since our results are concerned with algorithms for k-colourable graphs (for fixed values of k), we will consider only random models for k-colourable graphs. Two important issues guiding our choice of random models are: (1) the model should be amenable to probabilistic analysis and (2) the model should support distributions which are polynomial time realizable, so that we can compare and evaluate different colouring algorithms. Dyer and Frieze [3] describe several such models. They also proved that, under some assumptions, all these models are equivalent to one another in the sense that results derived for one model also hold for the other models. One standard model, called the partition model, ensures that the random graph is k-colourable by initially partitioning the vertex set into k colour classes. We consider one widely studied version of this model denoted by $\mathscr{G}(n,p,k)$. In the following, we assume $V = \{1,\ldots,n\}$.

$\mathscr{G}(n,p,k)$ model. Here, the vertex set V is partitioned into k colour classes $V_1,\ldots,V_k$. Usually, we also assume that the sizes of the colour classes are such that $|V_i| = n_i \geqslant n/C$ for all i, for some constant $C \geqslant k$. Then edges between vertices in different colour classes are included with probability p. Here, the edges are chosen independently.

Some authors have considered a slightly different model denoted $\mathscr{GUC}(n,p,k)$, which is the same as the $\mathscr{G}(\cdot)$ model except that initially each vertex is uniformly randomly put into one of the k colour classes, instead of being put arbitrarily into one of the colour classes as is done in the $\mathscr{G}(\cdot)$ model. However, with exponentially low failure probability, *every* colour class formed by the $\mathscr{GUC}(\cdot)$ model has size at least n/C for some constant $C \geqslant k$. Hence algorithms that succeed on the $\mathscr{G}(\cdot)$ model will also succeed on the $\mathscr{GUC}(\cdot)$ model. Both $\mathscr{G}(n,p,k)$ and $\mathscr{GUC}(\cdot)$ models suffer from some drawbacks. To overcome these drawbacks, random graphs were generalized to the notion of *semi-random* k-colourable graphs, and algorithms have been developed for such graphs. For more details, see [2, 17, 16].

1.2. Previous results on algorithms for colouring random graphs

For general random graphs where we do not bound the chromatic number, a.s. approximation algorithms have been proposed before ([7, 11, 12, 18]). As for random k-colourable graphs, we have the following results. For more algorithmic results on colouring random graphs, the reader is referred to the recent survey article by Frieze and McDiarmid [5]. When $p = p(n)$ is fixed, the a.s. algorithms given by Turner [18], Dyer and Frieze [3] succeed with very high probability. [2, 1] provide a.s. algorithms that succeed even when $p(n)$ goes to 0 very quickly.

The algorithms given in [3] work under the assumption that all colour classes have sizes between $[1 - o(1)]n/3$ and $[1 + o(1)]n/3$ and are proved to have failure probability at most e^{-n^δ} for some positive constant δ. However, using the techniques described in this paper, we can remove this assumption and can make these algorithms work for graphs drawn from $\mathscr{G}(n,p,k)$.

Blum and Spencer [2] give polynomial time algorithms for colouring graphs from the $\mathscr{GUC}(n,p(n),k)$ model where $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 0$. The main idea behind their algorithm goes like this. Given two vertices x and y, the algorithm computes the number of paths of length l (for a suitable positive integer l) in the input graph between x and y. Based on this number, it *concludes* that x and y belong to the same or different colour classes. Blum and Spencer [2] prove that this algorithm succeeds in k-colouring the input random graph with probability at least $1 - o(n^{-D})$ for some positive constant D.

Alon and Kahale [1] give polynomial time algorithms for colouring graphs from the $\mathscr{GUC}(n,p(n),k)$ model for $p(n)$ as low as $p(n) \geqslant c/n$ for some sufficiently large positive constant c. Their algorithms are based on the spectral properties of the input graph. They first compute the eigenvalues and corresponding eigenvectors of the adjacency matrix of a subgraph. Based on these, they compute an approximation to the colour classes and then refine it to obtain a k-colouring. Alon and Kahale [1] prove that the algorithms have polynomially low failure probability.

Dyer and Frieze [3] describe algorithms for k-colouring random graphs from $\mathcal{G}(n,p,k)$ (with p and k fixed), under the assumption mentioned before, in polynomial average running time. They show that the greedy colouring algorithm proposed by Kucera [11] has failure probability at most $e^{-n\cdot[1-o(1)]}$. When greedy colouring fails, one applies an intermediate algorithm Colour such that the failure probability of Colour is $e^{-n^{\beta}}$ for some constant $\beta > 1$. When Colour also fails, one resorts to brute-force colouring. Subramanian, Furer and Madhavan [16] describe p.av.t. algorithms for the more general semi-random model $\mathcal{G}_{SB}(n,p,k)$ with $p(n)$ going to 0 asymptotically. Their approach requires the failure probability of the underlying a.s. algorithm to be at most $e^{-(\log n)^3/p(n)}$ for $p \geqslant n^{-1+\epsilon}, \epsilon > 0$. Subramanian, Furer and Madhavan [16] also present another approach (for $p \geqslant n^{-\frac{2k}{(k-1)(k+2)}+\epsilon}$, $\epsilon > 0$) for colouring semi-random graphs in p.av.t., which only requires an a.s. algorithm with exponentially low failure probability, that is, at most $e^{-n^{\delta}}$ for some fixed $\delta > 0$.

1.3. Our contributions

We present the following algorithmic results on k-colouring random k-colourable graphs.

1. Almost surely succeeding polynomial time algorithms for k-colouring random graphs from $\mathcal{G}(n,p,k)$ with $p \geqslant n^{-1+\epsilon}$ where $\epsilon > 1/4$ is a constant.

Even though our results work only when $p \geqslant n^{-1+\epsilon}$, $\epsilon > 1/4$, they work for the stronger $\mathcal{G}(n,p,k)$ model as against the results of [2] and [1], which were meant for the $\mathcal{GUC}(n,p,k)$ model. Also, they are quite simple and are combinatorial in nature. In addition, our a.s. algorithms have the advantage that their failure probability is provably exponentially low. This is desirable when we try to design p.av.t. algorithms for k-colouring.

However, as pointed out in [16], colouring 'most' of the graphs does not always mean colouring a majority of relevant input graphs because what constitutes 'most' of the graphs depends on the distribution chosen and also how we define the term 'most'. There are often situations when we may not mind spending more time on some inputs but would like to have a k-colouring. Also, a.s. algorithms can have varying guarantees of failure probabilities such as $o((\log n)^{-\Omega(1)})$, $o(n^{-\Omega(1)})$, $o(n^{-\Omega(\log n)})$, $o(e^{-\Omega(n^{\delta})})$, and so on. Certainly, all these algorithms are not equally good. Designing a single a.s. algorithm provides only one guarantee of failure probability. Ideally, we would prefer to have algorithms which guarantee very low failure probabilities. A p.av.t. algorithm guarantees this by providing a continuous trade-off between running time and failure probabilities. In particular, we can achieve any desired polynomially low failure probability at the cost of polynomial increase in running time. For more details, we refer the reader to [16].

More importantly, aiming for p.av.t. algorithms forces us to design algorithms with very low failure probability. Our second major contribution is the following.

2. Polynomial average time algorithms for k-colouring random graphs from $\mathcal{G}(n,p,k)$ with $p \geqslant n^{-1+\epsilon}$ where $\epsilon > 1/4$ is a positive constant.

Our results on p.av.t. algorithms significantly improve the previously known results. Our results are based on three different techniques for designing p.av.t. algorithms using a.s. polynomial time algorithms. Each technique works by applying a series of intermediate algorithms with increasing running times and decreasing failure probabilities. These intermediate algorithms represent a trade-off between running time and failure probability. We can reduce failure probability to $O(e^{-\Omega(m \cdot n^{\epsilon})})$ if we are willing to spend $O(n^{O(m)})$ time on n-vertex graphs.† A similar approach has been used in [16] for obtaining p.av.t. algorithms for semi-random graphs, but these only work for a much smaller range of edge probabilities mentioned before. However, for the $\mathscr{G}(n, p, k)$ model, by exploiting the uniformity, we present techniques that work even for those a.s. algorithms whose failure probability is 'much higher' than that required by [3] or [16]. In fact, the second and third techniques only require exponentially low failure probability and work for $p \geqslant n^{-1+\epsilon}, \epsilon > 1/4$. As a result, we are able to k-colour sparse random graphs in polynomial average time.

The paper is organized as follows. In Section 2 we introduce some graph-theoretic notation and basic facts about asymptotic behaviour of functions. These will be used often in the rest of the paper. In Section 3 we present the a.s. algorithms for k-colouring random graphs. In Section 4 we present p.av.t. algorithms for k-colouring random graphs. Finally, in Section 5 we conclude with some remarks on open problems and future directions. The work presented here also appears in [15] and a part of the work has previously appeared in extended abstract form [13, 14].

2. Some preliminaries

2.1. Notation and notions

For a graph $G = (V, E)$ and any $S \subseteq V$ and $u \in V$, we use the following notation often.

- $N_S(u) = \{ v \in S \mid \text{u is adjacent to } v \}$; $d_S(u) = |N_S(u)|$. In particular, if $S = V$, then $N_V(u)$ is the set of neighbours of u in V, and $d_V(u)$ is the degree of u in V. We also denote the set $N_V(u)$ by $N(u)$ and $d_V(u)$ by $d(u)$. Also, for any set $T \subseteq V$, the neighbourhood $N(T)$ of T is defined by $N(T) = \{ u \in V \mid u \text{ is adjacent to some } v \in T \} = \bigcup_{v \in T} N(v)$.

We use the notation $G \in \mathscr{G}(n, p, k)$ to mean that G is a random k-colourable graph drawn from the model $\mathscr{G}(n, p, k)$. Throughout Sections 3 and 4, we let $V_1, \ldots, V_k$, with sizes $n_1, \ldots, n_k$, respectively, be the k colour classes used in the model. Assume $n_i \geqslant n/C$ for all i for some constant $C \geqslant k$. Let F denote the set of all potential edges between vertices in different colour classes. For all i, $1 \leqslant i \leqslant k$, let W_i denote the set $V_i \cup \ldots \cup V_k$ and G_i denote the subgraph of G induced by the set W_i. Also, for each i, let m_i denote the size of the set W_i, that is, $m_i = n_i + \cdots + n_k$. We also define M_i as $\sum_{i < l_1 < l_2 \leqslant k} n_{l_1} \cdot n_{l_2}$ for all $i = 1, \ldots, k-2$ and 0 for $i = k-1$. M_i is the maximum number of edges that G_{i+1} can have. Also, we are interested only in the asymptotic behaviour of our algorithms.

2.2. Some basic facts on probability estimates and asymptotic functions

We will very frequently use the Chernoff bounds (see [8]) on the tail probabilities of Bernoulli trials. We state and prove some simple facts about the asymptotic behaviour of certain functions. We also state some estimates on the probability of certain events

† There is a difference between the trade-offs mentioned in the previous and this paragraph. In the former, we obtain a continuous trade-off from an existing p.av.t. algorithm. In the latter, we provide a (not necessarily continuous) trade-off to get a p.av.t. algorithm.

occurring. We will often use these facts later. Let $f(n)$ be any real-valued function over positive integers. We use the notation $f(n) = o(1)$ to capture the fact that the function $f(n)$ tends to 0 as n tends to infinity. Often, we are only concerned with the asymptotic behaviour of a function, and not with any explicit description of the function. Hence we use the notation $o(1)$ to denote any function $f(n)$ such that $f(n) = o(1)$, without mentioning explicitly the function $f(n)$. The actual function $f(n)$ is to be inferred from the context. The equalities and inequalities in the following facts are understood to hold for sufficiently large n.

Fact 2.1. *Let $M(n)$, $p(n)$ be, respectively, nonnegative integer-valued and nonnegative real-valued functions such that $M(n) \to \infty$, $p(n) \to 0$ and $M(n)p(n) \to 0$ as $n \to \infty$. Then,*

$$(1 - p(n))^{M(n)} \quad = \quad 1 - M(n) \cdot p(n) \cdot [1 - o(1)]. \qquad \square$$

Fact 2.2. *Let $S(n)$ be a set of vertices such that $|S(n)| = M(n)$. Let u be any vertex that is not in $S(n)$. Let each edge $\{u, s\}$ $(s \in S(n))$ be selected with equal probability $p(n)$. If $M = M(n)$ and $p = p(n)$ are such that $M(n)p(n) = o(1)$ and $M(n) \to \infty$ as $n \to \infty$, then*

$$\begin{aligned}
&\Pr(u \textit{ is adjacent to some vertex in } S(n)) \\
&\qquad = \quad Mp - M^2p^2[1 - o(1)]/2 \;=\; Mp[1 - o(1)], \\
&\Pr(u \textit{ is adjacent to at least two vertices in } S(n)) \\
&\qquad = \quad M^2p^2[1 - o(1)]/2.
\end{aligned}$$

Proof. We have

$$\begin{aligned}
&\Pr(u \text{ is adjacent to some vertex in } S(n)) \\
&\qquad = \quad 1 - (1 - p(n))^{M(n)} \\
&\qquad = \quad Mp - M(M-1)p^2/2 + \text{higher order terms}, \\
&\Pr(u \text{ is adjacent to at least two vertices in } S(n)) \\
&\qquad = \quad 1 - (1-p)^M - Mp \cdot (1-p)^{M-1} \\
&\qquad = \quad Mp - M(M-1)p^2[1 - o(1)]/2 - Mp + M(M-1)p^2[1 - o(1)].
\end{aligned}$$

Using the assumptions $M(n)p(n) = o(1)$ and $M(n) \to \infty$, we have

$$\begin{aligned}
&\Pr(u \text{ is adjacent to some vertex in } S(n)) \\
&\qquad = \quad Mp - M^2p^2[1 - o(1)]/2 = M(n)p(n)[1 - o(1)], \\
&\Pr(u \text{ is adjacent to at least two vertices in } S(n)) \\
&\qquad = \quad M(n)^2p(n)^2[1 - o(1)]/2. \qquad \square
\end{aligned}$$

Fact 2.3. *Let α^* be any constant such that $0 < \alpha^* < 1$. Let f, g be any two constants such that $0 \leqslant f < g \leqslant 1$. Let α be any value (not necessarily a constant) such that $\alpha^* \leqslant \alpha < 1$.*

Let l, s be any two constant positive integers. Then we have

$$\begin{aligned}(1-f\alpha)^l \cdot [1-o(1)]^s &\geqslant (1-g\alpha)^l,\\ (1+g\alpha)^l \cdot [1-o(1)]^s &\geqslant (1+f\alpha)^l,\\ (1-g\alpha)^l \cdot [1+o(1)]^s &\leqslant (1-f\alpha)^l,\\ (1+f\alpha)^l \cdot [1+o(1)]^s &\leqslant (1+g\alpha)^l.\end{aligned}$$

Proof. We use the following two facts, which can be verified easily. For any constant positive integer s, we have $[1-o(1)]^s = [1-o(1)]$ and $[1+o(1)]^s = [1+o(1)]$. Also, we have $\alpha^* \leqslant \alpha < 1$, where α^* is a positive constant. Using these, we have

$$\begin{aligned}(1-f\alpha)^l \cdot [1-o(1)]^s &= (1-f\alpha)^l \cdot [1-o(1)]\\ &= (1-f\alpha)^l - (1-f\alpha)^l \cdot o(1)\\ &= (1-f\alpha)^l - o(1)\\ &\geqslant (1-g\alpha)^l.\end{aligned}$$

Similarly, we can derive the other inequalities. □

For any vertex x, and for any l $(1 \leqslant l \leqslant k)$, let $N_l(x)$ (or $n_l(x)$) denote the set of (or number of) neighbours of x in the colour class V_l. If y is any other vertex, then $N_l(xy)$ (or $n_l(xy)$) denotes $N_l(x) \cup N_l(y)$ (or $|N_l(x) \cup N_l(y)|$). For any vertex w, $w \neq x$, $w \neq y$, with each of the edges $\{x, w\}, \{y, w\}$ being chosen independently with probability p,

$$\Pr(w \text{ is adjacent to at least one of } x \text{ or } y) \quad = \quad p(2-p).$$

Let α^* be any positive constant. Using Chernoff bounds, we deduce that the following hold in the random graph G.

P1: For any l $(1 \leqslant l \leqslant k)$ and for any fixed $x \notin V_l$, we have, with probability at least $1 - e^{-\Omega(np)}$, the following inequality:

$$(1-\alpha^*) \cdot n_l \cdot p \quad \leqslant \quad n_l(x) \quad \leqslant \quad (1+\alpha^*) \cdot n_l \cdot p.$$

P2: For any l $(1 \leqslant l \leqslant k)$ and for any fixed $x, y \notin V_l$, we have, with probability at least $1 - e^{-\Omega(np)}$, the following inequality:

$$(1-\alpha^*) \cdot n_l \cdot p(2-p) \quad \leqslant \quad n_l(xy) \quad \leqslant \quad (1+\alpha^*) \cdot n_l \cdot p(2-p).$$

P3: If $(1/p(n))^2 = o(n)$, then for any l $(1 \leqslant l \leqslant k)$ and for any fixed $x, y \notin V_l$, we have, with probability at least $1 - e^{-\Omega(np^2)}$, the following inequality:

$$(1-\alpha^*) \cdot n_l \cdot p^2 \quad \leqslant \quad |N_l(x) \cap N_l(y)| \quad \leqslant \quad (1+\alpha^*) \cdot n_l \cdot p^2.$$

3. Almost surely succeeding algorithms for colouring random graphs

In this section, we present new a.s. algorithms for k-colouring the $\mathcal{G}(n, p(n), k)$ model for values of $p(n)$ such that $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 1/4$. In Section 3.1 we describe the common theme underlying all these new algorithms. In Sections 3.2 and 3.3 we present the new algorithms. Our algorithms have exponentially low failure probability. Precisely, the failure probability of our algorithms is at most $e^{-\Omega(\min(np,n^2p^3))}$ or at most $e^{-\Omega(\min(np,n^3p^4))}$, depending on the value of $p(n)$.

3.1. Outline of the algorithms for colouring random graphs

Our algorithms depend on the values of $p(n)$ and the sizes n_i, to compute a k-colouring of G; but we do not know these values. However, this is not a serious obstacle and this difficulty can be overcome, as is shown in the following theorem.

Theorem 3.1. *Let $A(G,n,k,(n'_1,\ldots,n'_k),q)$ be a deterministic algorithm for k-colouring the random graph G. Let $T_A(n)$ denote the maximum running time of algorithm A on any n-vertex graph. Assume that A succeeds on G with probability at least $1-o(1)$* if *$n'_i = n_i$ for all $i=1,\ldots,k$ and $q=p[1\pm o(1)]$. Then there exists another deterministic algorithm $B(G,n,k)$ such that B succeeds on G with probability at least $1-o(1)$ and the maximum running time of algorithm B on any n-vertex graph is $O(n^{k+1}\cdot T_A(n))$.*

Proof. The main idea behind algorithm B is to invoke $A(G,n,k,\ (n'_1,\ldots,n'_k),q)$ for all possible values of n'_i, $i=1,\ldots,k$. Algorithm B works as follows.

Algorithm $B(G,n,k)$.

```
  (1) $m(G)$ = number of edges in the input graph $G$
  (2) for all $(n'_1,\ldots,n'_k)$ such that $\sum_{1\leqslant i\leqslant k} n'_i = n$ and $n'_i \geqslant 0$ for all $i$ do
     (2a) $M = \sum_{1\leqslant i<j\leqslant k} n'_i \cdot n'_j$
          /* $M$ is the actual number of potential edges if $n'_i = n_i$ for all $i$ */
     (2b) $q(G) = m(G)/M$
     (2c) Apply $A(G,n,k,(n'_1,\ldots,n'_k),q(G))$; if $A$ succeeds, then exit.
     end /* of for loop */
end /* of algorithm $B(G,n,k)$ */
```

Clearly there are only $O(n^{k+1})$ solutions to the equation $\sum_{1\leqslant i\leqslant k} n'_i = n, \quad n'_i \geqslant 0$ for all i. Hence algorithm A will be called at most $O(n^{k+1})$ times in step (2c). Since A runs in $T_A(n)$ time, algorithm B runs in $O(n^{k+1}\cdot T_A(n))$ time. Consider the iteration of the *for* loop when we have $n'_i = n_i$ for all i where n_i are the sizes of the colour classes used by the adversary. We will surely come to this iteration, denoted *iter*, unless A has already succeeded in some previous iteration. We have $n_i \geqslant n/C$ for all i, for some constant $C \geqslant k$. Hence the value of M (defined in step (2a)) in this iteration is such that $M = \Omega(n^2)$. Hence, using Chernoff's bounds, for $\delta = 1/(\log n)$, we have

$$(1-\delta)\cdot Mp \quad\leqslant\quad m \quad\leqslant\quad (1+\delta)\cdot Mp$$

with probability at least $1-e^{-\Omega(Mp\delta^2)}$. In other words, we have $q(G) = m(G)/M = p[1\pm o(1)]$ with probability at least $1-e^{-n^{\beta}}$ for some constant $\beta > 1$. Hence

$$\begin{aligned}
&\Pr(\text{algorithm } B \text{ fails on } G)\\
&\qquad\leqslant \Pr(\text{algorithm } A \text{ fails on } G \text{ during } \textit{iter})\\
&\qquad\leqslant \Pr(\text{inequality (3.1) is not satisfied during } \textit{iter})\\
&\qquad\qquad + \Pr(\text{inequality (3.1) is satisfied, but algorithm } A \text{ fails on } G)\\
&\qquad\leqslant e^{-n^{\beta}} + o(1)\\
&\qquad= o(1).
\end{aligned}$$

Thus algorithm B succeeds with probability $1 - o(1)$. □

Note. Hence it is enough to design a polynomial time algorithm $A(G, n, k, (n'_1, \ldots, n'_k), q)$ such that A succeeds on G with probability $1 - o(1)$ whenever $n'_i = n_i$, for all i and $q = p[1 \pm o(1)]$. Also, we can assume that the algorithm knows the values of n_i for all i and hence knows the approximate value q of p.

The common idea behind the a.s. algorithms we are going to describe in this section is to separate the colour classes one by one. Suppose there exists some $x \in V_1$ and a polynomial time computable quantity $n(x, y, G)$ such that, for all elements y of V, we can correctly infer from $n(x, y, G)$ whether $y \in V_1$ or not. Then, by enumerating all possible choices of x, we can separate the colour class V_1. Now apply the same procedure to the remaining graph. This idea is formally stated and proved in the next theorem. As before, $q(G), m(G)$ have their usual meanings and M is the number of potential edges considered by the adversary.

Theorem 3.2. *Let $G \in \mathscr{G}(n, p(n), k)$, $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 0$ be a random graph. Let V_i, W_i, n_i, $G_i = G(W_i)$ have their usual meanings. Let $n(x, y, G)$ be a polynomial time computable function. Suppose there exist $k-1$ fixed vertices $x_i \in V_i$ and $k-1$ polynomial time computable values $B(n'_i, \ldots, n'_k, q(G))$ $(i = 1, \ldots, k-1)$ such that the condition* **Cond**(G) *(described below) holds with probability at least $1 - f(n)$ $(0 \leqslant f(n) \leqslant 1)$.*

Cond(G). *For all i $(1 \leqslant i \leqslant k-1)$, and for all $y \in W_i$, $y \neq x_i$,*

$$\begin{aligned}
n(x_i, y, G_i) &\geqslant B(n_i, \ldots, n_k, q(G)), && \textit{if } y \in V_i,\\
n(x_i, y, G_i) &< B(n_i, \ldots, n_k, q(G)), && \textit{if } y \in V_j,\ j > i.
\end{aligned}$$

Then there exists a polynomial time algorithm $B(G, n, k)$ such that B succeeds in k-colouring G with probability at least $1 - (f(n) + e^{-n \cdot \log n})$.

Proof. It is clear from the proof of Theorem 3.1 that it is enough to design a polynomial time algorithm ColourBySep$(G, n, k, (n'_1, \ldots, n'_k), q)$, given below, such that ColourBySep succeeds on G with probability $1 - f(n)$ whenever $n'_i = n_i$, for all i and $q = q(G) = p[1 \pm o(1)]$. Once we have got such an algorithm, the algorithm $B(G, n, k)$ can be constructed easily as explained in the proof of Theorem 3.1.

```
**ColourBySep**$(H, n, i, (n'_{k-i+1}, \ldots, n'_k), q)$.
  (1) if $i = 1$ then
    (2) if $H$ has no edge then return $V(H)$ /* success */
      /* else failure */
  (3) for all vertices $u \in V(H)$ do
    (4) $CC = \{u\} \cup \{\, v \in V(H) \mid n(u, v, H) \geqslant B(n'_{k-i+1}, \ldots, n'_k, q) \,\}$;
    (5) if $|CC|$ is an independent set of size $n'_{k-i+1}$ then
      (6) Apply ColourBySep$(H - CC, n, i-1, (n'_{k-i+2}, \ldots, n'_k), q)$.
      (7) if ColourBySep succeeds in obtaining a $(i-1)$-colouring $\{C_j\}$ of $H - CC$,
            then
      (7a) output the $i$-colouring $\{CC\} \cup \{C_j\}$ of $H$.
    end /* for loop */
end /* of ColourBySep */
```

It is clear from the description of the algorithm that, if G satisfies **Cond**(G) mentioned before, then ColourBySep succeeds in k-colouring G, whenever $n'_i = n_i$, for all i. Since, by assumption, G satisfies **Cond**(G) with probability at least $1 - f(n)$, we conclude that ColourBySep succeeds on G with probability at least $1 - f(n)$ whenever $n'_i = n_i$, for all i. Hence there exists a polynomial time algorithm $B(G, n, k)$ for k-colouring G which succeeds with probability at least $1 - (f(n) + e^{-n \cdot \log n})$. □

Remark 3.1. In the above theorem, we can also interchange the positions of the two inequalities ($\geqslant$ and $<$) appearing in the definition of **Cond**(G). That is, $n(x, y, G) < B(n_i, \ldots, n_k, q)$ if $y \in V_i$ and $n(x, y, G) \geqslant B(n_i, \ldots, n_k, q)$ if $y \in V_j,\ j > i$.

Remark 3.2. In the above theorem, $n(x, y, G)$ denotes a unique value of a polynomial time computable function. We can generalize the theorem by allowing $n(x, y, G)$ to be *any* value returned by a suitable polynomial time algorithm A. In this case, we require that **Cond**(G) holds true for *all* values returned by the algorithm A. The theorem also holds true in the general case.

3.2. Algorithms for the case of $p \geqslant n^{-1+\epsilon}$, $\epsilon > 1/3$

In this subsection, we prove the existence of a.s. algorithms for the case of $\epsilon > 1/3$. In view of the note given after Theorem 3.1, we can assume that our algorithms know the values of n_i and also the approximate value q of p. The algorithm works by removing the colour classes one by one in *decreasing* order of their sizes. It first removes the largest colour class, then the second largest colour class, and so on. Given two vertices x and y such that x belongs to the largest colour class, the algorithm determines whether y belongs to the largest colour class or not, by computing the number of edges in the subgraph of G induced by the set $N(x) \cup N(y)$.

Theorem 3.3. *Let $G \in \mathscr{G}(n, p(n), k)$, $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 1/3$ be a random graph. Then there exists a polynomial time algorithm A for k-colouring G such that A succeeds with probability at least $1 - e^{-\Omega(\min(np, n^2p^3))}$.*

Proof. As before, let $V_i, n_i, W_i, G_i, m_i, C, q(G), m(G)$ have their usual meanings. We assume, without loss of generality, that $n_1 \geqslant \ldots \geqslant n_k$. We have $q(G) = m(G)/M$, where $M = \sum_{1 \leqslant i < j \leqslant k} n_i n_j$ and $q = p[1 \pm o(1)]$ with probability at least $1 - e^{-n^{\beta}}$ for some constant $\beta > 1$.

For any two vertices x and y, let $n(x, y, G)$ denote the number of edges in the subgraph of G induced by the set $N(x) \cup N(y)$. Clearly, $n(x, y, G)$ is a polynomial time computable function. By Theorem 3.2, it suffices to prove the existence of $k-1$ fixed vertices $x_i \in V_i$ $(1 \leqslant i \leqslant k-1)$ and $k-1$ polynomial time computable quantities $B(n_i, \ldots, n_k, q)$ such that G satisfies **Cond**(G), with probability at least $1 - e^{-\Omega(\min(np, n^2 p^3))}$.

Compute the positive value α such that, for all $i = 1, \ldots, k-1$,

$$(1-\alpha)^3 \cdot n_i \cdot n_j \quad > \quad 8\alpha(3+\alpha^2) \cdot M_i, \quad \text{for all } j > i,$$

and at least one of these inequalities is violated if we use 2α instead of α. α can be computed in constant time from the values of $n_1, \ldots, n_k$. It is easy to verify, using $n_i \geqslant n/C$ for all i, that α exists and is bounded below by a *sufficiently small* positive constant α^*. Since α^* is assumed to be small, we have $\alpha \geqslant \alpha/4 \geqslant \alpha^*$. Let $\beta = \alpha/2$.

Define the quantities $B(n_i, \ldots, n_k, q)$ $(i = 1, \ldots, k-1)$ as follows:

$$B(n_i, \ldots, n_k, q) \quad = \quad (1-\beta)^3 \cdot q^3 \cdot (M_i \cdot (2-q)^2 + n_i \cdot n_k).$$

Consider any i $(i = 1, \ldots, k-1)$ and any *fixed* vertex $x_i \in V_i$. Let $y \in W_i$ be any other vertex. Assume that inequalities P1 and P2 hold. Given that this happens, using Facts 2.2 and 2.3 and inequalities P1 and P2 and the fact $q = p[1 \pm o(1)]$ and hence $p = q[1 \pm o(1)]$, and also the assumption $n_1 \geqslant \ldots \geqslant n_k$, we deduce that the following hold with probability at least $1 - e^{-\Omega(n^2 p^3)}$:

$$\begin{aligned}
y \in V_i \quad : \quad n(x_i, y, G_i) \quad &\leqslant \quad (1+\alpha^*)^3 \cdot p^3 \cdot (2-p)^2 \cdot M_i \\
&\leqslant \quad (1+\alpha/4)^3 \cdot q^3 \cdot (2-q)^2 \cdot [1+o(1)]^5 \cdot M_i \\
&\leqslant \quad (1+\alpha/2)^3 \cdot q^3 \cdot (2-q)^2 \cdot M_i \\
&= \quad (1+\beta)^3 \cdot q^3 \cdot (2-q)^2 \cdot M_i \\
&= \quad (1-\beta)^3 \cdot q^3 \cdot (M_i \cdot (2-q)^2 + n_i n_k) \\
&\quad\;\; - \; q^3 \cdot \left(n_i n_k (1-\beta)^3 - 2\beta(3+\beta^2) \cdot (2-q)^2 \cdot M_i\right) \\
&< \quad B(n_i, \ldots, n_k, q).
\end{aligned}$$

The choice of $\beta = \alpha/2$ helps us in obtaining the strict inequality $<$ in the last line. Also,

$$\begin{aligned}
y \in V_j, j > i \quad : \quad n(x_i, y, G_i) \quad \geqslant \quad & (1-\alpha^*)^3 \cdot p^3 \cdot (X(i,j)(2-p)^2 \\
& + Y(i,j)(2-p) + Z(i,j)(2-p) + n_i n_j),
\end{aligned}$$

where

$$\begin{aligned}
X(i,j) \quad &= \sum_{i+1 \leqslant l_1 < l_2 \leqslant k,\ l_1 \neq j,\ l_2 \neq j} n_{l_1} \cdot n_{l_2}, \\
Y(i,j) \quad &= \sum_{i+1 \leqslant l \leqslant k,\ l \neq j} n_j \cdot n_l, \\
Z(i,j) \quad &= \sum_{i+1 \leqslant l \leqslant k,\ l \neq j} n_i \cdot n_l.
\end{aligned}$$

By assumption, we have $n_i \geqslant n_j$. Using this, we derive that

$$X(i,j)\cdot(2-p)^2 + Y(i,j)\cdot(2-p) + Z(i,j)\cdot(2-p) \quad \geqslant \quad M_i\cdot(2-p)^2.$$

Hence, we have

$$\begin{aligned} n(x_i,y,G_i) &\geqslant (1-\alpha^*)^3\cdot p^3\cdot(M_i\cdot(2-p)^2+n_i\cdot n_j) \\ &\geqslant (1-\alpha/4)^3\cdot p^3\cdot(M_i\cdot(2-p)^2+n_i\cdot n_j) \\ &\geqslant (1-\alpha/2)^3\cdot q^3\cdot(M_i\cdot(2-q)^2+n_i\cdot n_j) \\ &= (1-\beta)^3\cdot q^3\cdot(M_i\cdot(2-q)^2+n_i\cdot n_j) \\ &\geqslant B(n_i,\dots,n_k,q). \end{aligned}$$

Now, for a given x_i, inequalities P1 and P2 hold for $y \neq x_i$, with probability at least $1-e^{-\Omega(np)}$. Thus, with probability at least $1-e^{-\Omega(\min(np,n^2p^3))}$, the random graph G satisfies **Cond**(G) with inequalities interchanged in their positions. Hence, by Remark 3.1, there exists a polynomial time algorithm $B(G,n,k)$ that k-colours G with probability at least $1-e^{-\Omega(\min(np,n^2p^3))}$. □

3.3. Algorithms for the case of $p \geqslant n^{-1+\epsilon}$, $\epsilon > 1/4$

We first design a.s. algorithms for the subcase of $p(n) \leqslant n^{-1+\epsilon_2}$ where $\epsilon_2 < 1/2$ is a constant. Then we extend this to $p(n) \geqslant n^{-1+\epsilon}$ where $\epsilon > 1/4$ is a constant. The algorithm works by removing the colour classes one by one in *increasing* order of their sizes. Given two vertices x and y such that x belongs to the smallest colour class, the algorithm determines whether y belongs to the smallest colour class or not, by computing the number of vertices in G which are adjacent to at least two vertices in $N(x)\cup N(y)$.

Theorem 3.4. *Let $G \in \mathcal{G}(n,p(n),k)$ be a random graph with $p(n) \geqslant n^{-1+\epsilon_1}$ and $p(n) \leqslant n^{-1+\epsilon_2}$ where ϵ_1 and ϵ_2 are positive constants such that $1/4 < \epsilon_1 < \epsilon_2 < 1/2$. Then there exists a polynomial time algorithm A for k-colouring G such that A succeeds with probability at least $1-e^{-\Omega(\min(np,n^3p^4))}$.*

Proof. As before, let $V_i, n_i, C, W_i, G_i, m_i, C, q(G), m(G), M$ have their usual meanings. We assume, without loss of generality, that $n_1 \leqslant \ldots \leqslant n_k$. For any two vertices x and y, let $n(x,y,G)$ be the size of the set

$$\{v \notin N_G(x)\cup N_G(y) : v \text{ is adjacent to at least two vertices in } N_G(x)\cup N_G(y)\}.$$

Clearly, $n(x,y,G)$ is a polynomial time computable function. By Theorem 3.2, it suffices to prove the existence of $k-1$ fixed vertices $x_i \in V_i$ $(1 \leqslant i \leqslant k-1)$ and $k-1$ polynomial time computable quantities $B(n_i,\dots,n_k,q)$ such that G satisfies **Cond**(G) mentioned before, with probability at least $1-e^{-\Omega(\min(np,n^3p^4))}$. For arbitrary S and $w \notin S$, let $g(p,i,|S|)$ $(1 \leqslant i \leqslant |S|)$ denote the probability that w is adjacent to *at least* i vertices of S, assuming that each edge $\{w,s\}$ $(s \in S)$ is chosen with probability p.

Compute the positive value α such that, for all $i = 1,\dots,k-1$,

$$n_j\cdot n_j\cdot n_i \quad > \quad 4\alpha(3+\alpha^2)\cdot\left((m_i-n_i)^2n_i+\sum_{r>i}(m_i-n_i-n_r)^2n_r\right), \text{ for all } j>i,$$

and at least one of these inequalities is violated if we use 2α instead of α. α can be computed in constant time from the values of $n_1, \ldots, n_k$. Again, using $n_i \geqslant n/C$ for all i, α exists and is bounded below by a *sufficiently small* positive constant α^*. Since α^* is assumed to be small, we have $\alpha \geqslant \alpha/4 \geqslant \alpha^*$. Let $\beta = \alpha/2$.

Define the quantities $B(n_i, \ldots, n_k, q)$ $(i = 1, \ldots, k-1)$ as follows:

$$B(n_i, \ldots, n_k, q) \quad = \quad 2(1-\beta)^3 \cdot q^4 \cdot \left((m_i - n_i)^2 n_i + \sum_{r>i} (m_i - n_i - n_r)^2 n_r \right).$$

Consider any i $(i = 1, \ldots, k-1)$ and any *fixed* vertex $x_i \in V_i$. Let $y \in W_i$ be any other vertex. Now, with probability at least $1 - e^{-\Omega(np)}$, inequalities P1 and P2 hold. Assume that inequalities P1 and P2 hold for x_i and y. Also, since $p(n) \leqslant n^{-1+\epsilon_2}$ and $\epsilon_2 < 1/2$, we have $np^2 = o(1)$. Hence, for any $w \in V_r$ $(r \geqslant i)$,

$$\Pr\left(w \text{ is adjacent to at least 2 vertices of } \cup_{l \geqslant i, l \neq r} N_l(x_i, y)\right) \text{ (denote this by } q_r)$$

$$= g\left(p, 2, \sum_{l \geqslant i, l \neq r} n_l(x_i, y)\right) = \left(\sum_{l \geqslant i, l \neq r} n_l(x_i, y)\right)^2 \cdot p^2 \cdot [1 - o(1)]/2, \quad \text{using Fact 2.2.}$$

Write $n(x_i, y, G_i) = \sum_{r \geqslant i} n(x_i, y, G_i)_r$ where

$$n(x_i, y, G_i)_r = |\{v \in V_r \ : \ v \text{ is adjacent to at least two vertices in } \cup_{l \geqslant i, l \neq r} N_l(x_i, y)\}|.$$

Given that inequalities P1 and P2 hold for x_i and y, we prove that, with probability at least $1 - e^{-\Omega(n^3 p^4)}$, we have $n(x_i, y, G_i) \geqslant B(n_i, \ldots, n_k, q)$ if $y \in V_i$, and $n(x_i, y, G_i) < B(n_i, \ldots, n_k, q)$ if $y \notin V_i$. In proving this assertion, we often use Facts 2.2 and 2.3 and the fact $q = p[1 \pm o(1)]$ and hence $p = q[1 \pm o(1)]$, and also the assumption $n_1 \leqslant \ldots \leqslant n_k$.

Case $y \in V_i$. We have

$$\begin{aligned} q_i &= \left(\sum_{l>i} n_l(x_i, y)\right)^2 \cdot p^2 \cdot [1 - o(1)]/2 \\ &\geqslant (1-\alpha^*)^2 \cdot (m_i - n_i)^2 \cdot p^4 \cdot (2-p)^2 \cdot [1 - o(1)]/2. \end{aligned}$$

Hence

$$n(x_i, y, G_i)_i \quad \geqslant \quad (1-\alpha^*)^3 \cdot (m_i - n_i)^2 \cdot n_i \cdot p^4 \cdot (2-p)^2 \cdot [1 - o(1)]/2.$$

Similarly, for $r > i$,

$$\begin{aligned} q_r &= \left(\sum_{l>i, l \neq r} n_l(x_i, y)\right)^2 \cdot p^2 \cdot [1 - o(1)]/2 \\ &\geqslant (1-\alpha^*)^2 \cdot (m_i - n_i - n_r)^2 \cdot p^4 \cdot (2-p)^2 \cdot [1 - o(1)]/2. \end{aligned}$$

Hence, for $r > i$, we have

$$n(x_i, y, G_i)_r \quad \geqslant \quad (1-\alpha^*)^3 \cdot (m_i - n_i - n_r)^2 \cdot n_r \cdot p^4 \cdot (2-p)^2 \cdot [1 - o(1)]/2.$$

Hence,

$$\begin{aligned}
&n(x_i, y, G_i)\\
&\quad = \sum_{r \geqslant i} n(x_i, y, G_i)_r\\
&\quad \geqslant (1-\alpha^*)^3 p^4 (2-p)^2 [1-o(1)]/2 \cdot \left((m_i - n_i)^2 n_i + \sum_{r>i} (m_i - n_i - n_r)^2 n_r \right)\\
&\quad \geqslant (1-\beta)^3 \cdot q^4 \cdot 2 \cdot \left((m_i - n_i)^2 n_i + \sum_{r>i} (m_i - n_i - n_r)^2 n_r \right)\\
&\quad = B(n_i, \ldots, n_k, q).
\end{aligned}$$

Case $y \in V_j$, $j > i$. We have

$$\begin{aligned}
q_i &= \left(\sum_{l>i} n_l(x_i, y) \right)^2 \cdot p^2 \cdot [1-o(1)]/2\\
&\leqslant (1+\alpha^*)^2 \cdot p^4 \cdot ((m_i - n_i - n_j)(2-p) + n_j)^2/2,\\
q_j &= \left(\sum_{l \geqslant i, l \neq j} n_l(x_i, y) \right)^2 \cdot p^2 \cdot [1-o(1)]/2\\
&\leqslant (1+\alpha^*)^2 \cdot p^4 \cdot ((m_i - n_i - n_j)(2-p) + n_i)^2/2.
\end{aligned}$$

Hence,

$$\begin{aligned}
n(x_i, y, G_i)_i &\leqslant (1+\alpha^*)^3 \cdot p^4 \cdot ((m_i - n_i - n_j)(2-p) + n_j)^2 \cdot n_i/2,\\
n(x_i, y, G_i)_j &\leqslant (1+\alpha^*)^3 \cdot p^4 \cdot ((m_i - n_i - n_j)(2-p) + n_i)^2 \cdot n_j/2.
\end{aligned}$$

Similarly, for $r > i, r \neq j$,

$$\begin{aligned}
q_r &= \left(\sum_{l \geqslant i, l \neq r} n_l(x_i, y) \right)^2 \cdot p^2 \cdot [1-o(1)]/2\\
&\leqslant (1+\alpha^*)^2 \cdot p^4 \cdot ((m_i - n_i - n_j - n_r)(2-p) + n_i + n_j)^2/2.
\end{aligned}$$

Hence, for $r > i, r \neq j$, we have

$$\begin{aligned}
n(x_i, y, G_i)_r &\leqslant (1+\alpha^*)^3 \cdot p^4 \cdot ((m_i - n_i - n_j - n_r)(2-p) + n_i + n_j)^2 \cdot n_r/2\\
&\leqslant (1+\alpha^*)^3 \cdot p^4 \cdot ((m_i - n_i - n_r)(2-p) + n_j p)^2 \cdot n_r/2, \quad \text{using } n_i \leqslant n_j,\\
&\leqslant (1+\alpha^*)^3 \cdot p^4 \cdot (m_i - n_i - n_r)^2 \cdot (2-p)^2 [1+o(1)]^2 \cdot n_r/2\\
&\leqslant (1+\alpha/4)^3 \cdot q^4 \cdot (m_i - n_i - n_r)^2 \cdot 2 \cdot [1+o(1)]^6 \cdot n_r\\
&\leqslant (1+\beta)^3 \cdot q^4 \cdot (m_i - n_i - n_r)^2 \cdot n_r \cdot 2\\
&\leqslant (1-\beta)^3 \cdot q^4 \cdot (m_i - n_i - n_r)^2 \cdot n_r \cdot 2\\
&\quad + 4\beta(3+\beta^2) \cdot q^4 \cdot (m_i - n_i - n_r)^2 \cdot n_r.
\end{aligned}$$

Let

$$\begin{aligned} X(i,j) &= ((m_i-n_i-n_j)(2-p)+n_j)^2\cdot n_i \\ &\leqslant 4(m_i-n_i)^2\cdot n_i + n_j^2\cdot n_i - 4(m_i-n_i)\cdot n_i n_j, \\ Y(i,j) &= ((m_i-n_i-n_j)(2-p)+n_i)^2\cdot n_j \\ &\leqslant 4(m_i-n_i-n_j)^2\cdot n_j + n_i^2\cdot n_j + 4(m_i-n_i-n_j)\cdot n_i n_j. \end{aligned}$$

Hence, using $n_i \leqslant n_j$,

$$X(i,j)+Y(i,j) \leqslant 4(m_i-n_i)^2\cdot n_i + 4(m_i-n_i-n_j)^2\cdot n_j - 2n_j^2 n_i.$$

As a result,

$$\begin{aligned} & n(x_i,y,G_i)_i + n(x_i,y,G_i)_j \\ &\leqslant (1+\alpha^*)^3\cdot p^4\cdot (X(i,j)+Y(i,j))/2 \\ &\leqslant (1+\alpha^*)^3\cdot q^4\cdot 2\cdot((m_i-n_i)^2\cdot n_i + (m_i-n_i-n_j)^2\cdot n_j - n_j^2 n_i/2)[1+o(1)]^4 \\ &\leqslant (1+\beta)^3\cdot q^4\cdot 2\cdot((m_i-n_i)^2\cdot n_i + (m_i-n_i-n_j)^2\cdot n_j - n_j^2 n_i/2) \\ &\leqslant (1-\beta)^3\cdot q^4\cdot 2\cdot((m_i-n_i)^2\cdot n_i + (m_i-n_i-n_j)^2\cdot n_j) - (1+\beta)^3\cdot q^4\cdot n_j^2 n_i \\ &\quad + 4\beta(3+\beta^2)\cdot q^4\cdot((m_i-n_i)^2\cdot n_i + (m_i-n_i-n_j)^2\cdot n_j). \end{aligned}$$

Hence,

$$\begin{aligned} n(x_i,y,G_i) &= \sum_{r\geqslant i} n(x_i,y,G_i)_r \\ &\leqslant B(n_i,\ldots,n_k,q) - (1+\beta)^3\cdot q^4\cdot n_j^2\cdot n_i \\ &\quad + 4\beta(3+\beta^2)\cdot q^4\cdot\left((m_i-n_i)^2\cdot n_i + \sum_{r>i}(m_i-n_i-n_r)^2\cdot n_r\right) \\ &< B(n_i,\ldots,n_k,q). \end{aligned}$$

The last inequality follows from the choice of β. Now, for a given x_i and y, inequalities P1 and P2 hold for x_i and y with probability at least $1-e^{-\Omega(np)}$. Given that this happens, with probability at least $1-e^{-\Omega(\min(np,n^3p^4))}$, the random graph G satisfies **Cond**(G) for y. Hence, by Theorem 3.2, there exists a polynomial time algorithm $B(G,n,k)$ which k-colours G with probability at least $1-e^{-\Omega(\min(np,n^3p^4))}$. □

Using the previous theorem, we can design a.s. algorithms for the case of $p(n)\geqslant n^{-1+\epsilon}$, where ϵ is any constant greater than $1/4$. The idea is to apply both polynomial time algorithms for the cases of $p(n)\geqslant n^{-3/5}$ (Section 3.2) and $n^{-1+\epsilon}\leqslant p(n)\leqslant n^{-3/5}$ (Section 3.3). Since these two ranges of $p(n)$ are overlapping, at least one of the two algorithms will almost surely succeed. Hence we have the following theorem.

Theorem 3.5. *Let $G\in\mathscr{G}(n,p(n),k)$ be a random graph with $p(n)\geqslant n^{-1+\epsilon}$ where ϵ is any positive constant such that $\epsilon>1/4$. Then there exists a polynomial time algorithm A for k-colouring G such that A succeeds with probability at least $1-e^{-\Omega(\min(np,n^3p^4))}$ if $p(n)\leqslant n^{-1+2/5}$ and with probability at least $1-e^{-\Omega(\min(np,n^2p^3))}$ if $p(n)\geqslant n^{-1+2/5}$.* □

4. Polynomial average time algorithms for colouring random graphs

Here, we give p.av.t. algorithms for k-colouring random graphs from the $\mathcal{G}(n, p(n), k)$ model for $p(n)$ as low as $p(n) \geqslant n^{-1+\epsilon}$ where ϵ is any positive constant greater than $1/4$. Our algorithms are based on a collection of different techniques.

4.1. The central ideas

We generalize the idea used in [3] further. This approach has also been used to design p.av.t. algorithms for k-colouring semi-random graphs (see [16]). Let A be a polynomial time algorithm for k-colouring random graphs such that A succeeds with probability $1 - o(1)$. We introduce a sequence $\langle A_1, \ldots, A_r \rangle$ of one or more algorithms as intermediate steps between the polynomial time algorithm A and the brute-force colouring method. Each algorithm A_i is applied when all of the previous intermediate steps $A_j, j < i$, have failed. The running times $T(A_i, n)$ and failure probabilities $p_f(A_i, n)$ of successive algorithms in the sequence are such that, for all i, $T(A_i) = o(T(A_{i+1}))$ and $p_f(A_{i+1}) = o(p_f(A_i))$. We make sure that this idea works by properly choosing the sequence of intermediate algorithms.

We present three different types of intermediate algorithms. Using these, we present three different techniques for colouring random graphs in polynomial average time. There is a common theme underlying the design of these intermediate algorithms. We first describe this common theme in the next subsection. Then, in the subsequent subsections, we present the three types of intermediate algorithms and show how they lead to p.av.t. algorithms.

4.2. Outline of the intermediate algorithms

The different types of intermediate algorithms will all be referred to by the common name FindColour(G, n, k, m). Each type of FindColour(G, n, k, m) takes $O(n^{3 \cdot k \cdot m})$ time. If m satisfies certain constraints and $p(n)$ lies in a suitable range (which vary for different types of FindColour), then FindColour(G, n, k, m) succeeds with probability at least $1 - e^{-\Omega(m \cdot n^{\delta})}$ for some positive constant δ. In view of Theorem 3.1, it is enough to design an algorithm ColSepSubset$(G, n, k, (n'_1, \ldots, n'_k), m, q)$ such that

C1: ColSepSubset takes $O(n^{(2m+2) \cdot k})$ time, and

C2: if $n'_i = n_i$ for all $i = 1, \ldots, k$ and $q = p[1 \pm o(1)]$, then ColSepSubset succeeds in k-colouring G with probability at least $1 - e^{-\Omega(m \cdot n^{\delta})}$ for some positive constant δ. Here, as usual, $n_1, \ldots, n_k$ are the sizes of the k colour classes.

The idea behind the algorithm ColSepSubset is to separate the colour classes one by one and it is similar to the idea used in Section 3. Suppose there exists some $A \subseteq V_1, |A| = m$, and a polynomial time computable quantity $n(A, x, G)$ such that, for all but *at most* $2m$ elements of V, we can correctly infer from $n(A, x, G)$ whether $x \in V_1$ or not. Then, by enumerating all possible m-element subsets of V, we can separate the colour class V_1. Now apply the same procedure to the remaining graph. This idea is formally stated and proved in the next theorem.

Theorem 4.1. *Let $G \in \mathcal{G}(n, p(n), k)$, $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 0$ be a random graph. Let $V_i, W_i, n_i, G_i = G(W_i), m(G), q(G), M$ have their usual meanings. Let m be an integer such that $1 \leqslant m \leqslant n$. For an independent set A, and a vertex $y \notin A$, let $n(A, y, G)$ be a polynomial time computable function. Suppose there exist $k-1$ fixed subsets $A_i \subseteq V_i$ and $k-1$ polynomial time computable values $B(n'_i, \ldots, n'_k, q(G), m)$ $(i = 1, \ldots, k-1)$ such that the condition* **Cond**(G) *(described below) holds with probability at least $1 - f(n)$ $(0 \leqslant f(n) \leqslant 1)$.*

Cond(G). *For all i $(1 \leqslant i \leqslant k-1)$,*

for all but at most *m vertices $y \in V_i - A_i$,*

we have $n(A_i, y, G_i) \geqslant B(n_i, \ldots, n_k, q(G), m)$;

for all but at most *m vertices $y \in W_i - V_i$,*

we have $n(A_i, y, G_i) < B(n_i, \ldots, n_k, q(G), m)$.

Then there exists an algorithm FindColour(G, n, k, m) *such that* FindColour *takes $O(n^{3 \cdot k \cdot m})$ time and it succeeds in k-colouring G with probability at least $1 - (f(n) + e^{-n \cdot \log n})$.*

Proof. As explained before, it is enough to design an algorithm ColSepSubset$(G, n, k, (n'_1, \ldots, n'_k), q, m)$ such that ColSepSubset satisfies conditions C1 and C2 with $q = q(G) = p[1 \pm o(1)]$. Once we have such an algorithm, the algorithm FindColour(G, n, k, m) can be constructed easily as explained in the proof of Theorem 3.1. ColSepSubsetis given below.

ColSepSubset($H, n, i, (n'_{k-i+1}, \ldots, n'_k), q, m$).

```
(1) if $i = 1$ then
   (2) if $H$ has no edge then return $V(H)$ /* success */
      /* else failure */
(3) for all independent subsets $A \subseteq V(H)$ such that $|A| = m$ do
   (4) $CC = \{\, v \in V(H) \mid n(A, v, H) \geqslant B(n'_{k-i+1}, \ldots, n'_k, q, m) \,\}$;
   (5) for all subsets $Y_1 \subseteq CC$ and $Y_2 \subseteq V(H) - CC$ of size at most $m$ each do
      (6) if $(CC \cup Y_2) - Y_1$ is an independent set of size $n'_{k-i+1}$ then
      (6a) $CC = (CC \cup Y_2) - Y_1$;
      (6b) Apply ColSepSubset$(H - CC, n, i-1, (n'_{k-i+2}, \ldots, n'_k), q, m)$.
      (6c) if ColSepSubset succeeds in obtaining a $(i-1)$-colouring $\{C_j\}$ of $H - CC$,
            then
      (6d) output the $i$-colouring $\{CC\} \cup \{C_j\}$ of $H$.
      end /* for loop */
   end /* for loop */
end /* of ColSepSubset */
```

It is clear from the description of the algorithm that, if G satisfies **Cond**(G) and if $n'_i = n_i$ for all i and $q = q(G)$, then ColSepSubset succeeds in k-colouring G. By assumption, G satisfies **Cond**(G) with probability at least $1 - f(n)$. Also, $q(G) = p[1 \pm o(1)]$ with probability at least $1 - e^{-n \cdot (\log n)}$. Hence, by Theorem 3.1, there exists an algorithm FindColour(G, n, k, m) for k-colouring G which succeeds with probability at least $1 - (f(n) + e^{-n \cdot (\log n)})$. Also, it is clear from its description that ColSepSubset takes $O(n^{(2m+2)k})$ time and FindColour(G, n, k, m) takes $O(n^{3km})$ time. □

Remark 4.1. In the above theorem we can interchange the positions of the two inequalities ($\geqslant$ and $<$) appearing in the definition of **Cond**(G). That is, $n(A_i, y, G) < B(n_i, \ldots, n_k, q, m)$ for all *but at most* m vertices of V_i and $n(A_i, y, G) \geqslant B(n_i, \ldots, n_k, q, m)$ for all *but at most* m vertices of V_j, $j > i$

We choose three different types of polynomial time computable functions $n(A, y, G)$ yielding three different types of FindColour, namely, FindColour1, FindColour2 and FindColour3. In the following subsections we describe these and analyse their failure probabilities. Before that, we explain some notation and inequalities which will be used frequently in the rest of this section. As before, using the note given after Theorem 3.1, we can assume that all of our algorithms know the values of n_i for all i and also the value of q.

Additional notation and facts. For any set A of size m, $1 \leqslant m \leqslant n$, for any l, $1 \leqslant l \leqslant k$, let $N_l(A) = \{ v \in V_l \ : \ v \text{ is adjacent to some vertex of } A \}$; $n_l(A) = |N_l(A)|$. Also, for any $A \subseteq V_i$, $w \notin V_i$, $\Pr(w \text{ is adjacent to some vertex of } A) = 1 - (1-p)^{|A|}$.

Let α^* be any positive constant. Using Chernoff bounds, and also Fact 2.2, we deduce that the following hold in the random graph G.

P4: For any fixed independent set A of size m such that $m \cdot (\log n) \leqslant (1/p(n))$, and for any $l = 1, \ldots, k$ such that $A \cap V_l = \emptyset$, we have, with probability at least $1 - e^{-\Omega(mnp)}$,

$$(1-\alpha^*) \cdot m \cdot n_l \cdot p \quad \leqslant \quad n_l(A) \quad \leqslant \quad (1+\alpha^*) \cdot m \cdot n_l \cdot p.$$

P5: Let A be any fixed independent set of size m such that $m \cdot (\log n) \leqslant (1/p(n))$ and $y \notin A$ be any fixed vertex. Then, *given that* A satisfies inequality P4, we have for any $l = 1, \ldots, k$ such that $y \notin V_l$ and $A \cap V_l = \emptyset$, with probability at least $1 - e^{-\Omega(np)}$,

$$(1-\alpha^*) \cdot n_l \cdot p \quad \leqslant \quad |N_l(y) - N_l(A)| \quad \leqslant \quad (1+\alpha^*) \cdot n_l \cdot p.$$

4.3. Intermediate algorithm FindColour1(G, n, k, m)

Theorem 4.2. *Let $G \in \mathscr{G}(n, p(n), k)$, $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 0$. Let γ_2 be any positive constant. Then there exists a positive constant δ (depending on only ϵ and γ_2) and an algorithm* FindColour1(G, n, k, m) $(1 \leqslant m \leqslant n)$ *running in $O(n^{3 \cdot k \cdot m})$ time such that,*

if $m \cdot (\log n) \leqslant 1/p(n)$ and $n^{\gamma_2} \leqslant m \cdot n \cdot p(n)^2$, then FindColour1$(G, n, k, m)$ *succeeds in k-colouring G with probability at least $1 - e^{-\Omega(m \cdot n^{\delta})}$.*

Proof. Let m be any integer satisfying the conditions mentioned above. Let δ be the constant defined by $\delta = \min(\gamma_2, \epsilon)$. For an arbitrary independent set A of size m and a vertex y such that $y \notin A$, let $n(A, y, G)$ denote the number of neighbours of y in $N_G(A)$. Clearly, given A and y, the quantity $n(A, y, G)$ is polynomial time computable. By Theorem 4.1, it suffices to prove the existence of $k-1$ fixed subsets $A_i \subseteq V_i$ $(1 \leqslant i \leqslant$

$k-1$) and $k-1$ quantities $B(n_i,\ldots,n_k,q,m)$ such that G satisfies **Cond**(G) mentioned in Theorem 4.1, with probability at least $1-e^{-\Omega(m\cdot n^{\delta})}$.

Compute the positive value α such that, for all $i=1,\ldots,k-1$,

$$(1-\alpha)^2\cdot n_j \quad > \quad 4\alpha\cdot(m_i-n_i-n_j), \quad \text{for all } j>i,$$

and at least one inequality is violated if we use 2α instead of α. α can be computed in constant time from the values of $n_1,\ldots,n_k$. Also, using $n_i\geqslant n/C$ for all i, α is bounded below by a *sufficiently small* positive constant α^*. Since α^* is assumed to be small, we have $\alpha\geqslant\alpha/4\geqslant\alpha^*$. Let $\beta=\alpha/2$.

Define the quantities $B(n_i,\ldots,n_k,q,m)$ $(i=1,\ldots,k-1)$ as follows:

$$B(n_i,\ldots,n_k,q,m) \quad = \quad (1-\beta)^2\cdot q^2\cdot m\cdot(m_i-n_i).$$

Consider any i $(i=1,\ldots,k-1)$ and any *fixed* subset $A_i\subseteq V_i$. Let $y\in W_i-A_i$ be any vertex. Assume that inequality P4 holds. Given that this happens, using Fact 2.3 and the fact $q=p[1\pm o(1)]$ and also the assumption $n^{\gamma_2}\leqslant m\cdot n\cdot p^2$, we deduce that the following hold with probability at least $1-e^{-\Omega(n^{\gamma_2})}$. If $y\in V_i$, then

$$\begin{aligned} n(A_i,y,G_i) &\geqslant (1-\alpha^*)^2\cdot p^2\cdot(m_i-n_i)\cdot m\\ &\geqslant (1-\beta)^2\cdot q^2\cdot(m_i-n_i)\cdot m\\ &\geqslant B(n_i,\ldots,n_k,q,m).\end{aligned}$$

If $y\in V_j$, where $j>i$, then

$$\begin{aligned} n(A_i,y,G_i) &\leqslant (1+\alpha^*)^2\cdot p^2\cdot(m_i-n_i-n_j)\cdot m\\ &\leqslant (1+\beta)^2\cdot q^2\cdot(m_i-n_i-n_j)\cdot m\\ &\leqslant (1-\beta)^2\cdot q^2\cdot(m_i-n_i)\cdot m\\ &\quad +4\beta\cdot q^2\cdot(m_i-n_i-n_j)\cdot m\\ &\quad -(1-\beta)^2\cdot q^2\cdot n_j\cdot m\\ &< B(n_i,\ldots,n_k,q,m) \quad \text{by the choice of } \beta.\end{aligned}$$

Thus, for any $y\in V_i$, $\Pr(n(A_i,y,G_i)\geqslant B(n_i,\ldots,n_k,q,m))\leqslant e^{-\Omega(n^{\gamma_2})}$. These probabilities are independent for different vertices of V_i. Hence, with probability at least $1-e^{-\Omega(m\cdot n^{\gamma_2})}$, there are *at most* m vertices of V_i-A_i that violate the inequality mentioned in the description of **Cond**(G). Similarly, with probability at least $1-e^{-\Omega(m\cdot n^{\gamma_2})}$, there are at most m vertices of W_i-V_i that violate the corresponding inequality in **Cond**(G). Now, for a given fixed $A_i\subseteq V_i$ of size m, inequality P4 holds with probability at least $1-e^{-\Omega(m\cdot n^{\epsilon})}$. Hence, with probability at least $1-e^{-\Omega(m\cdot n^{\delta})}$, the random graph G satisfies **Cond**(G). Hence, by Theorem 4.1, there exists an algorithm FindColour1(G,n,k) which k-colours G with probability at least $1-e^{-\Omega(m\cdot n^{\delta})}$. □

4.4. Intermediate algorithm FindColour2(G,n,k,m)

Theorem 4.3. *Let $G\in\mathcal{G}(n,p(n),k)$, $p(n)\geqslant n^{-1+\epsilon}$, $\epsilon>1/3$. Then there exists a positive constant δ (depending only on ϵ) and an algorithm* FindColour2(G,n,k,m) $(1\leqslant m\leqslant n)$ *running in $O(n^{3\cdot k\cdot m})$ time such that,* *if* $m \cdot (\log n) \leqslant 1/p(n)$, *then* FindColour2$(G, n, k, m)$ *succeeds in* k*-colouring* G *with probability at least* $1 - e^{-\Omega(m \cdot n^{\delta})}$.

Proof. Assume, without loss of generality, that $n_1 \geqslant \ldots \geqslant n_k$. Let m be any integer satisfying the conditions mentioned above. Let δ be the constant defined by $\delta = \min(\epsilon, -1 + 3\epsilon)$. For an arbitrary independent set A of size m and a vertex y such that $y \notin A$, let $n(A, y, G)$ denote the number of edges of the form $\{u, v\}$ where $u \in N_G(A)$ and $v \in N_G(y) - N_G(A)$. Clearly, given A and y, the quantity $n(A, y, G)$ is polynomial time computable. By Theorem 4.1, it suffices to prove the existence of $k-1$ fixed subsets $A_i \subseteq V_i$ $(1 \leqslant i \leqslant k-1)$ and $k-1$ quantities $B(n_i, \ldots, n_k, q, m)$ such that G satisfies **Cond**(G) mentioned in Theorem 4.1, with probability at least $1 - e^{-\Omega(m \cdot n^{\delta})}$.

Compute the positive value α such that, for all $i = 1, \ldots, k-1$,

$$(1-\alpha)^3 \cdot n_i \cdot n_j \quad > \quad 2\alpha(3+\alpha^2) \cdot M_i, \quad \text{for all } j > i,$$

and at least one inequality is violated if we use 2α instead of α. α can be computed in constant time from the values of $n_1, \ldots, n_k$. Again, $\alpha \geqslant \alpha^*$ for some *sufficiently small* positive constant α^*. Since α^* is assumed to be small, we have $\alpha \geqslant \alpha/4 \geqslant \alpha^*$. Let $\beta = \alpha/2$.

Define the quantities $B(n_i, \ldots, n_k, q, m)$ $(i = 1, \ldots, k-1)$ as follows:

$$B(n_i, \ldots, n_k, q, m) \quad = \quad (1-\beta)^3 \cdot q^3 \cdot m \cdot (M_i + n_i n_k).$$

Consider any i $(i = 1, \ldots, k-1)$ and any *fixed* subset $A_i \subseteq V_i$. Let $y \in W_i - A_i$ be any vertex. Assume that inequality P4 and P5 are satisfied by A and y. Given that this happens, using Fact 2.3 and the fact $q = p[1 \pm o(1)]$, we deduce that the following hold with probability at least $1 - e^{-\Omega(m \cdot n^2 \cdot p^3)}$. If $y \in V_i$, then

$$\begin{aligned} n(A_i, y, G_i) &\leqslant (1+\alpha^*)^3 \cdot p^3 \cdot m \cdot M_i \\ &\leqslant (1+\beta)^3 \cdot q^3 \cdot m \cdot M_i \\ &\leqslant B(n_i, \ldots, n_k, q, m) - q^3 \cdot m \cdot \left((1-\beta)^3 \cdot n_i \cdot n_k - 2\beta(3+\beta^2) \cdot M_i\right) \\ &< B(n_i, \ldots, n_k, q, m). \end{aligned}$$

If $y \in V_j$, where $j > i$, then

$$n(A_i, y, G_i) \quad \geqslant \quad (1-\alpha^*)^3 \cdot p^3 \cdot m \cdot (X(i,j) + Y(i,j) + Z(i,j) + U(i,j)),$$

where

$$\begin{aligned} X(i,j) &= \sum_{i<l_1<l_2 \leqslant k,\ l_1, l_2 \neq j} n_{l_1} \cdot n_{l_2}, \\ Y(i,j) &= \sum_{i<l \leqslant k, l \neq j} n_l \cdot n_j, \\ Z(i,j) &= \sum_{i<l \leqslant k, l \neq j} n_l \cdot n_i, \\ U(i,j) &= n_i \cdot n_j. \end{aligned}$$

Using $n_i \geqslant n_j \geqslant n_k$, we deduce that $X(i,j)+Y(i,j)+Z(i,j)+U(i,j) \geqslant M_i + n_i \cdot n_k$. Hence we have

$$\begin{aligned} n(A_i,y,G_i) &\geqslant (1-\alpha^*)^3 \cdot p^3 \cdot m \cdot (M_i + n_i \cdot n_k) \\ &\geqslant (1-\beta)^3 \cdot q^3 \cdot m \cdot (M_i + n_i \cdot n_k) \\ &\geqslant B(n_i,\ldots,n_k,q,m). \end{aligned}$$

Thus, given that A_i satisfies inequality P4, we have with probability at least $1-e^{-\Omega(m \cdot n^2 \cdot p^3)}$ that $n(A_i,y,G_i) \geqslant B(n_i,\ldots,n_k,q,m)$ (or $n(A_i,y,G_i) < B(n_i,\ldots,n_k,q,m)$) if $y \notin V_i$ (or $y \in V_i$), *for every* $y \in W_i - A_i$ such that y satisfies inequality P5. Now, given that A_i satisfies inequality P4, with probability at least $1-e^{-\Omega(m \cdot np)}$, there are *at most* m vertices that do not satisfy inequality P5. Also, A_i satisfies inequality P4 with probability at least $1-e^{-\Omega(mnp)}$. Hence, with probability at least $1-e^{-\Omega(m \cdot n^\delta)}$, the random graph G satisfies **Cond**(G). Hence, by Theorem 4.1, there exists an algorithm FindColour2(G,n,k) that k-colours G with probability at least $1-e^{-\Omega(m \cdot n^\delta)}$. □

4.5. Intermediate algorithm FindColour3(G,n,k,m)

Theorem 4.4. *Let $G \in \mathcal{G}(n,p(n),k)$, with $n^{-1+\epsilon_1} \leqslant p(n) \leqslant n^{-1+\epsilon_2}$ where ϵ_1 and ϵ_2 are positive constants such that $1/4 < \epsilon_1 < \epsilon_2 < 1/2$. Then there exists a positive constant δ (depending on only ϵ_1) and an algorithm* FindColour3(G,n,k,m) $(1 \leqslant m \leqslant n)$ *running in $O(n^{3 \cdot k \cdot m})$ time such that,*

> *if $m \cdot n \cdot p^2 \leqslant 1/(\log n)$, then* FindColour3$(G,n,k,m)$ *succeeds in k-colouring G with probability at least $1-e^{-\Omega(m \cdot n^\delta)}$.*

Proof. Assume, without loss of generality, that $n_1 \leqslant \ldots \leqslant n_k$. Let m be any integer satisfying the conditions mentioned above. Let δ be the constant defined by $\delta = \min(\epsilon_1, -1+4\epsilon_1)$. For an arbitrary independent set A of size m and a vertex y such that $y \notin A$, let $n(A,y,G)$ denote the number of vertices $u \notin A \cup \{y\}$ such that u is adjacent to some vertex of $N_G(A)$ *and* some vertex of $N_G(y) - N_G(A)$. Clearly, given A and y, the quantity $n(A,y,G)$ is polynomial time computable. By Theorem 4.1, it suffices to prove the existence of $k-1$ fixed subsets $A_i \subseteq V_i$ $(1 \leqslant i \leqslant k-1)$ and $k-1$ quantities $B(n_i,\ldots,n_k,q,m)$ such that G satisfies **Cond**(G) mentioned before, with probability at least $1-e^{-\Omega(m \cdot n^\delta)}$.

Compute the positive value α such that, for all $i = 1,\ldots,k-1$,

$$n_j \cdot n_j \cdot n_i \quad > \quad 2\alpha(3+\alpha^2) \cdot \left((m_i - n_i)^2 n_i + \sum_{r>i} (m_i - n_i - n_r)^2 n_r \right), \text{ for all } j > i,$$

and at least one inequality is violated if we use 2α instead of α. α can be computed in constant time from the values of $n_1,\ldots,n_k$. Also, $\alpha \geqslant \alpha^*$ for some *sufficiently small* positive constant α^*. Since α^* is assumed to be small, we have $\alpha \geqslant \alpha/4 \geqslant \alpha^*$. Let $\beta = \alpha/2$.

Define the quantities $B(n_i,\ldots,n_k,q,m)$ $(i = 1,\ldots,k-1)$ as follows:

$$B(n_i,\ldots,n_k,q,m) \quad = \quad (1-\beta)^3 \cdot q^4 \cdot m \cdot \left((m_i - n_i)^2 n_i + \left(\sum_{r>i} (m_i - n_i - n_r)^2 n_r \right) \right).$$

Consider any i ($i = 1, \ldots, k-1$) and any *fixed* subset $A_i \subseteq V_i$. Let $y \in W_i - A_i$ be any vertex. Assume that inequalities P4 and P5 are satisfied by A and y.

Now, since $p(n) \leqslant n^{-1+\epsilon_2}$ and $\epsilon_2 < 1/2$, we have $np^2 = o(1)$. Hence, for any $w \in V_r$ ($r \geqslant i$), we have that

$$\begin{aligned}&\Pr\big(w \text{ is adjacent to some vertex of } N_{G_i}(y) - N_{G_i}(A_i)\big) \quad (\text{denote this by } q_r(y, A_i))\\ &\qquad = \left(\sum_{l \geqslant i, l \neq j, l \neq r} |N_l(y) - N_l(A_i)|\right) \cdot p \cdot [1 - o(1)], \quad \text{using Fact 2.2.}\end{aligned}$$

Also, since, by assumption, $m \cdot n \cdot p^2 \leqslant 1/(\log n)$ we have $m \cdot n \cdot p^2 = o(1)$. Hence, for any $w \in V_r$ ($r \geqslant i$), we have that

$$\begin{aligned}&\Pr\big(w \text{ is adjacent to some vertex of } N_{G_i}(A_i)\big) \quad (\text{denote this by } q_r(A_i))\\ &\qquad = \left(\sum_{l > i, l \neq r} |N_l(A_i)|\right) \cdot p \cdot [1 - o(1)], \quad \text{using Fact 2.2.}\end{aligned}$$

Write $n(A_i, y, G_i) = \sum_{l \geqslant i} n(A_i, y, G_i)_l$, where

$$\begin{aligned} n(A_i, y, G_i)_l \;\; = \;\; & |\{v \in V_l \;:\; v \text{ is adjacent to some vertex in } N_{G_i}(A_i) \\ & \text{and some vertex in } N_{G_i}(y) - N_{G_i}(A_i)\}|.\end{aligned}$$

Now, using Fact 2.2 and the fact $q = p[1 \pm o(1)]$, and also the assumption $n_1 \leqslant \ldots \leqslant n_k$, we deduce the following.

Case $y \in V_i$. We have

$$\begin{aligned} q_i(y, A_i) &= \left(\sum_{l > i} |N_l(y) - N_l(A_i)|\right) \cdot p \cdot [1 - o(1)] \\ &\geqslant (1 - \alpha^*) \cdot (m_i - n_i) \cdot p^2 \cdot [1 - o(1)], \\ q_i(A_i) &= \left(\sum_{l > i} |N_l(A_i)|\right) \cdot p \cdot [1 - o(1)] \\ &\geqslant (1 - \alpha^*) \cdot (m_i - n_i) \cdot m \cdot p^2 \cdot [1 - o(1)]. \end{aligned}$$

Hence, with probability at least $1 - e^{-\Omega(m \cdot n^3 p^4)}$,

$$n(A_i, y, G_i)_i \;\geqslant\; (1 - \alpha^*)^3 \cdot (m_i - n_i)^2 \cdot m \cdot p^4 \cdot n_i \cdot [1 - o(1)].$$

Similarly, for $r > i$,

$$\begin{aligned} q_r(y, A_i) &= \left(\sum_{l > i, l \neq r} |N_l(y) - N_l(A_i)|\right) \cdot p \cdot [1 - o(1)] \\ &\geqslant (1 - \alpha^*) \cdot (m_i - n_i - n_r) \cdot p^2 \cdot [1 - o(1)], \\ q_r(A_i) &= \left(\sum_{l > i, l \neq r} |N_l(A_i)|\right) \cdot p \cdot [1 - o(1)] \\ &\geqslant (1 - \alpha^*) \cdot (m_i - n_i - n_r) \cdot m \cdot p^2 \cdot [1 - o(1)]. \end{aligned}$$

Hence, for $r > i$, with probability at least $1 - e^{-\Omega(m \cdot n^3 p^4)}$, we have

$$n(A_i, y, G_i)_r \geqslant (1-\alpha^*)^3 \cdot (m_i - n_i - n_r)^2 \cdot m \cdot p^4 \cdot n_r \cdot [1 - o(1)].$$

Hence

$$\begin{aligned} n(A_i, y, G_i) & \\ &= \sum_{r \geqslant i} n(A_i, y, G_i)_r \\ &\geqslant (1-\alpha^*)^3 \cdot m \cdot p^4 \cdot [1-o(1)] \cdot \left((m_i - n_i)^2 n_i + \sum_{r>i} (m_i - n_i - n_r)^2 n_r \right) \\ &\geqslant (1-\beta)^3 \cdot m \cdot q^4 \cdot \left((m_i - n_i)^2 n_i + \sum_{r>i} (m_i - n_i - n_r)^2 n_r \right) \\ &= B(n_i, \ldots, n_k, q, m). \end{aligned}$$

Case $y \in V_j$, $j > i$. We have

$$\begin{aligned} q_i(y, A_i) &= \left(\sum_{l>i, l \neq j} |N_l(y) - N_l(A_i)| \right) \cdot p \cdot [1 - o(1)] \\ &\leqslant (1+\alpha^*) \cdot (m_i - n_i - n_j) \cdot p^2, \\ q_i(A_i) &= \left(\sum_{l>i} |N_l(A_i)| \right) \cdot p \cdot [1 - o(1)] \\ &\leqslant (1+\alpha^*) \cdot (m_i - n_i) \cdot m \cdot p^2, \\ q_j(y, A_i) &= \left(\sum_{l \geqslant i, l \neq j} |N_l(y) - N_l(A_i)| \right) \cdot p \cdot [1 - o(1)] \\ &\leqslant (1+\alpha^*) \cdot (m_i - n_j) \cdot p^2, \\ q_j(A_i) &= \left(\sum_{l>i, l \neq j} |N_l(A_i)| \right) \cdot p \cdot [1 - o(1)] \\ &\leqslant (1+\alpha^*) \cdot (m_i - n_i - n_j) \cdot m \cdot p^2. \end{aligned}$$

Hence, with probability at least $1 - e^{-\Omega(m \cdot n^3 p^4)}$,

$$\begin{aligned} n(A_i, y, G_i)_i &\leqslant (1+\alpha^*)^3 \cdot m \cdot p^4 \cdot (m_i - n_i) \cdot (m_i - n_i - n_j) \cdot n_i, \\ n(A_i, y, G_i)_j &\leqslant (1+\alpha^*)^3 \cdot m \cdot p^4 \cdot (m_i - n_j) \cdot (m_i - n_i - n_j) \cdot n_j. \end{aligned}$$

Similarly, for $r > i, r \neq j$,

$$\begin{aligned} q_r(y, A_i) &= \left(\sum_{l \geqslant i, l \neq j, l \neq r} |N_l(y) - N_l(A_i)| \right) \cdot p \cdot [1 - o(1)] \\ &\leqslant (1+\alpha^*) \cdot (m_i - n_j - n_r) \cdot p^2, \end{aligned}$$

$$\begin{aligned} q_r(A_i) &= \left(\sum_{l>i, l \neq r} |N_l(A_i)| \right) \cdot p \cdot [1 - o(1)] \\ &\leqslant (1 + \alpha^*) \cdot (m_i - n_i - n_r) \cdot m \cdot p^2. \end{aligned}$$

Hence, for $r > i, r \neq j$, using $n_i \leqslant n_j$, we have

$$\begin{aligned} n(A_i, y, G_i)_r &\leqslant (1 + \alpha^*)^3 \cdot m \cdot p^4 \cdot (m_i - n_i - n_r) \cdot (m_i - n_j - n_r) \cdot n_r \\ &\leqslant (1 + \alpha^*)^3 \cdot m \cdot p^4 \cdot (m_i - n_i - n_r)^2 \cdot n_r \\ &\leqslant (1 + \beta)^3 \cdot m \cdot q^4 \cdot (m_i - n_i - n_r)^2 \cdot n_r \\ &\leqslant (1 - \beta)^3 \cdot m \cdot q^4 \cdot (m_i - n_i - n_r)^2 \cdot n_r \\ &\quad + 2\beta(3 + \beta^2) \cdot m \cdot q^4 \cdot (m_i - n_i - n_r)^2 \cdot n_r. \end{aligned}$$

Let

$$\begin{aligned} X(i, j) &= (m_i - n_i) \cdot (m_i - n_i - n_j) \cdot n_i \\ &= (m_i - n_i)^2 \cdot n_i - (m_i - n_i) \cdot n_j \cdot n_i, \\ Y(i, j) &= (m_i - n_j) \cdot (m_i - n_i - n_j) \cdot n_j \\ &= (m_i - n_i - n_j)^2 \cdot n_j + (m_i - n_i - n_j) \cdot n_i \cdot n_j. \end{aligned}$$

Hence,

$$X(i, j) + Y(i, j) = (m_i - n_i)^2 \cdot n_i + (m_i - n_i - n_j)^2 \cdot n_j - n_j^2 \cdot n_i.$$

As a result,

$$\begin{aligned} & n(A_i, y, G_i)_i + n(A_i, y, G_i)_j \\ & \leqslant (1 + \alpha^*)^3 \cdot m \cdot p^4 \cdot (X(i, j) + Y(i, j)) \\ & \leqslant (1 + \beta)^3 \cdot m \cdot q^4 \cdot (X(i, j) + Y(i, j)) \\ & \leqslant (1 - \beta)^3 \cdot m \cdot q^4 \cdot \left((m_i - n_i)^2 \cdot n_i + (m_i - n_i - n_j)^2 \cdot n_j\right) - (1 + \beta)^3 \cdot m \cdot q^4 \cdot n_j^2 n_i \\ & \quad + 2\beta(3 + \beta^2) \cdot m \cdot q^4 \cdot \left((m_i - n_i)^2 \cdot n_i + (m_i - n_i - n_j)^2 \cdot n_j\right). \end{aligned}$$

Hence,

$$\begin{aligned} n(A_i, y, G_i) &= \sum_{r \geqslant i} n(A_i, y, G_i)_r \\ &\leqslant B(n_i, \ldots, n_k, q, m) - (1 + \beta)^3 \cdot m \cdot q^4 \cdot n_j^2 \cdot n_i \\ &\quad + 2\beta(3 + \beta^2) \cdot m \cdot q^4 \cdot \left((m_i - n_i)^2 \cdot n_i + \sum_{r>i} (m_i - n_i - n_r)^2 \cdot n_r \right) \\ &< B(n_i, \ldots, n_k, q, m). \end{aligned}$$

The last inequality follows from the choice of β. Thus, given that A_i satisfies inequality P4, we have with probability at least $1 - e^{-\Omega(m \cdot n^3 \cdot p^4)}$ that $n(A_i, y, G_i) \geqslant B(n_i, \ldots, n_k, q, m)$ (or $n(A_i, y, G_i) < B(n_i, \ldots, n_k, q, m)$) if $y \in V_i$ (or $y \notin V_i$), *for every* $y \in W_i - A_i$ such that y satisfies inequality P5. Now, given that A_i satisfies inequality P4, with probability at least $1 - e^{-\Omega(mnp)}$, there are *at most* m vertices that do not satisfy inequality P5. Also, A_i satisfies inequality P4 with probability at least $1 - e^{-\Omega(mnp)}$. Hence, with probability at least

$1 - e^{-\Omega(m \cdot n^{\delta})}$, the random graph G satisfies **Cond**(G). Hence, by Theorem 4.1, there exists an algorithm FindColour3(G, n, k) that k-colours G with probability at least $1 - e^{-\Omega(m \cdot n^{\delta})}$.

□

Remark 4.2. For each of Theorems 4.2, 4.3 and 4.4, its proof actually guarantees larger success probability than is given in its statement:

(i) FindColour1 succeeds with probability at least $1 - e^{-\Omega(m^2 np^2)}$;
(ii) FindColour2 succeeds with probability at least $1 - e^{-\Omega(\min(mnp, mn^2p^3))}$;
(iii) FindColour3 succeeds with probability at least $1 - e^{-\Omega(\min(mnp, mn^3p^4))}$.

We have given the probabilities in terms of a constant δ so that the theorems become more convenient to use in the analysis of p.av.t. algorithms that follow.

In the following subsections, we show how to use the three FindColour() algorithms to design three different techniques for constructing p.av.t. algorithms for k-colouring $\mathcal{G}(n, p(n), k)$. One intermediate algorithm, called Colour2(G, n, k, m, m), has already been introduced in [16] to k-colour semi-random graphs in p.av.t. It also works for random graphs, since random graphs form a special class of semi-random graphs. Each of our techniques for p.av.t. algorithms works by using Colour2() and some FindColour(). For a description of Colour2(), see [16]. We will only be using the following facts about Colour2. It runs in $O(n^{3km})$ time and has failure probability at most e^{-nk} for $m \geqslant (\log n \cdot \log\log n)/p(n)$. Also, the following theorem from [16] regarding Colour2() will be used in our proofs.

Theorem 4.5. *Let $G \in \mathcal{G}(n, p(n), k)$ where $p(n) \geqslant n^{-1+\epsilon}$ for some positive constant ϵ. Let A be any polynomial time (worst-case) deterministic algorithm that k-colours G with probability at least $1 - e^{-(\log n)^3/p(n)}$. Then we can construct another algorithm $B(G, n, k)$ which always k-colours G and whose running time is polynomial on average.*

4.6. First technique for designing polynomial average time algorithms

Each of the algorithms FindColour?(G, n, k, m) succeeds only when m satisfies the corresponding inequality involving m and $p(n)$. But the only restriction we impose on $p(n)$ is that $p \geqslant n^{-1+\epsilon}$. So, p can vary between as high as a constant value and as low as $n^{-1+\epsilon}$ even for successive values of n. This leads to some difficulties in analysis. However, we overcome this difficulty by dividing the range $[n^{-1+\epsilon}, 1]$ into smaller subranges and then proving the result for each smaller subrange.

Theorem 4.6. *Let $G \in \mathcal{G}(n, p(n), k)$, $p = p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 0$ be a random graph. Let A be a deterministic polynomial time (worst-case) algorithm such that A succeeds in k-colouring G with probability at least $1 - e^{-\Omega(n^{\gamma} \cdot \max(1, 1/(np^2)))}$, where γ is some positive constant. Then there exists another deterministic algorithm $B1(G, n, k)$ which always k-colours G and whose running time is polynomial on average.*

Proof. The algorithm $B1(G, n, k)$ works as follows.

```
(1) Apply algorithm $A$ over the input random graph $G$; **if** $A$ succeeds, **then** exit.
(2) **for** $m = 1$ *to* $n/(\log n)^3$ **do**
      (2a) Apply Colour2$(G, n, k, m, m)$; **if** Colour2 succeeds, **then** exit.
      (2b) Apply FindColour1$(G, n, k, m)$; **if** FindColour1 succeeds, **then** exit.
(3) Apply brute-force colouring method over $G$.
```

It is clear that algorithm $B1$ always k-colours G. Now we prove that algorithm $B1(G, n, k)$ has polynomial average running time *for all sufficiently large values of* n.

Let τ be a *sufficiently small* constant. Divide the interval $[\epsilon, 1]$ into subintervals $[\epsilon + l\tau, \min(1, \epsilon + (l+1)\tau)]$ $(l \geqslant 0)$. Now, using this division of $[\epsilon, 1]$, divide the range $[n^{-1+\epsilon}, 1]$ into subranges $I_l = [n^{-1+\epsilon+l\tau}, \min(1, n^{-1+\epsilon+(l+1)\tau})]$, $l \geqslant 0$. Since ϵ and τ are constants, there is only a constant number of subranges I_l. Hence it is enough to prove that algorithm $B1(G, n, k)$ has polynomial running time for all sufficiently large values of n, as long as $p(n) \in I_l$ for some $l \geqslant 0$. Hence, in the rest of the proof we assume that $p(n) \in I_l$ for some $l \geqslant 0$. Let $p_{\text{low}} = n^{-1+\epsilon+l\tau}$ and $p_{\text{up}} = \min(1, n^{-1+\epsilon+(l+1)\tau})$. Using Theorem 4.5, without loss of generality, we assume that $p(n)$ is not in the last subrange corresponding to $p_{\text{up}} = 1$.

Let $f(n,p) = n^{\gamma} \cdot \max(1, 1/(np^2))$. Since $p \in I_l$, we have $f(n,p) \geqslant f_{\text{low}}(n,p)$ where $f_{\text{low}}(n,p) = n^{\gamma} \cdot \max(1, n^{1-2\epsilon-2(l+1)\tau})$ and $f(n,p) \leqslant f_{\text{up}}(n,p)$ where $f_{\text{up}}(n,p) = n^{\gamma} \cdot \max(1, n^{1-2\epsilon-2l\tau})$. We also have $f_{\text{low}}(n,p) \geqslant n^{\gamma}$. For all m such that $m \geqslant f_{\text{low}}(n,p)/(n^{\tau} \cdot (\log n)^6)$, we have $m \cdot n \cdot p^2 \geqslant n^{\gamma_2}$ for some constant $\gamma_2 > 0$. Let $\delta = \min(\epsilon, \gamma_2)$ be the constant mentioned in the statement and proof of Theorem 4.2 and let $\delta_1 = \delta/2$. Since τ is sufficiently small, we have $\delta_1 > \tau$.

First, we note that, if $f_{\text{low}}(n,p) \geqslant (\log n)^3/p_{\text{low}}$, then we can apply Theorem 4.5 and deduce that, for all sufficiently large values of n, $B1(G, n, k)$ has polynomial average running time. Hence, in the rest of the proof we assume that $f_{\text{low}}(n,p) \leqslant (\log n)^3/p_{\text{low}}$.

We use the following notation.

- Let $m_1(n)$ be a function defined as $m_1(n) = \lceil f_{\text{low}}(n,p)/(n^{\tau} \cdot (\log n)^6) \rceil$. Let $n_f = n/(\log n)^3$.
- Define $\beta_0 = 0$. Let r be a nonnegative *integer constant* and let $\beta_1 \leqslant \ldots \leqslant \beta_{r+1}$ be a sequence of *positive real constants* such that

  (i) $0 < \beta_{j+1} - \beta_j \leqslant \delta_1$, for all $j \leqslant r$;

  (ii) $m_1(n) \cdot n^{\beta_r} \cdot (\log n)^3 \leqslant 1/p_{\text{up}}$ and $(1/p_{\text{low}}) \cdot (\log n)^3 \leqslant m_1(n) \cdot n^{\beta_{r+1}}$.

  Since δ_1 and τ are constants, and since $\tau < \delta_1$, such constants r and $\{\beta_j\}$ always exist. The reason for choosing such constants is the following argument. FindColour1 (G, n, k, m) is useful for the analysis only when m 'lies' between the two barriers $1/(np^2)$ and $1/p$. Also, Colour2(G, n, k, m, m) is useful for the analysis only when m is 'above' the barrier $1/p$ [16].
- For $0 \leqslant j \leqslant r+1$, define $n_j = m_1(n) \cdot n^{\beta_j}$.
- Let $T_A(n)$, $T_{bf}(n)$, $T_i(n)$ denote the worst-case running times of algorithms A, brute-force colouring and the ith iteration of step (2), respectively. $E(T_{B1}(n))$ denotes the average running time of algorithm $B1$ over G.
- $q(A) = \Pr(A$ fails on $G)$. We have $q(A) \leqslant e^{-\Omega(f(n,p))} \leqslant e^{-\Omega(f_{\text{low}}(n,p))}$.
- For all m, $q(A, m)$ denotes the probability that algorithm A fails on G and for each $m' < m$, the m'th iteration of step (2) fails to obtain a k-colouring of G.

From the earlier results we have the following.

- For all values of m such that $m \geqslant m_1(n)$ and $m \leqslant n_r$, FindColour1(G, n, k, m) succeeds on G with probability at least $1 - e^{-\Omega(m \cdot n^{\delta})}$. Hence, for $j \leqslant r$, we have $q(A, n_j + 1) \leqslant e^{-n_j \cdot n^{\delta}}$.
- For all values of m such that $m \geqslant n_{r+1}$, Colour2(G, n, k, m, m) succeeds in k-colouring G with probability at least $1 - e^{-nk}$ [16]. Hence, for $j > r$, we have $q(A, n_j + 1) \leqslant e^{-nk}$.

Hence we have

$$\begin{aligned} E(T_{B1}(n)) &\leqslant T_A(n) + \sum_{1 \leqslant m \leqslant n_f} \left(\sum_{1 \leqslant m' \leqslant m} T_{m'}(n) \right) \cdot q(A, m) \\ &\quad + \left(T_{bf}(n) + \sum_{1 \leqslant m' \leqslant n_f} T_{m'}(n) \right) \cdot q(A, n_f + 1), \end{aligned}$$

$$\begin{aligned} E(T_{B1}(n)) &\leqslant T_A(n) + \left[\sum_{1 \leqslant m \leqslant n_0} m \cdot T_m(n) \right] \cdot q(A) \\ &\quad + \sum_{1 \leqslant j \leqslant r+1} \left[\sum_{n_{j-1} < m \leqslant n_j} m \cdot T_m(n) \right] \cdot q(A, n_{j-1} + 1) \\ &\quad + \left[\sum_{n_{r+1} < m \leqslant n_f} m \cdot T_m(n) + T_{bf}(n) \right] \cdot q(A, n_{r+1} + 1), \end{aligned}$$

$$\begin{aligned} E(T_{B1}(n)) &\leqslant T_A(n) + O(n_0^2 \cdot n^{3 \cdot k \cdot n_0}) \cdot e^{-\Omega(f_{\text{low}}(n,p))} \\ &\quad + \sum_{1 \leqslant j \leqslant r+1} O(n_j^2 \cdot n^{3 \cdot k \cdot n_j}) \cdot e^{-n_{j-1} \cdot n^{\delta}} + O(n^2 k^n) \cdot e^{-nk}. \end{aligned}$$

Substituting the values of n_j for various values of j, $1 \leqslant j \leqslant r + 1$, we can verify that each of the last three terms on the right-hand side of the inequality is less than 1. Hence we have $E(T_{B1}(n)) \leqslant 2 \cdot T_A(n)$. Thus the algorithm $B1$ has polynomial expected running time. Also, algorithm $B1$ always k-colours G since the brute-force colouring method always succeeds on G. This completes the proof of the theorem. □

4.7. Second technique for designing polynomial average time algorithms

In this subsection, we describe the second technique for designing p.av.t. algorithms for colouring random graphs. It works by repeatedly calling Colour2 and FindColour2. Unlike the first technique, this technique works for much higher failure probabilities, namely any exponentially low failure probability. However, it works only for a smaller range of edge probabilities, namely $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 1/3$.

Theorem 4.7. *Let $G \in \mathcal{G}(n, p(n), k)$, $p = p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 1/3$ be a random graph. Let A be a deterministic polynomial time (worst-case) algorithm such that A succeeds in k-colouring G with probability at least $1 - e^{-\Omega(n^{\gamma})}$, where γ is some positive constant. Then there exists a deterministic algorithm $B2(G, n, k)$ which always k-colours G and which runs in polynomial average time.*

Proof. The algorithm $B2(G,n,k)$ is the same as $B1(G,n,k)$ except that we apply FindColour2(G,n,k,m) instead of FindColour1() in step (2b).

It is clear that algorithm $B2$ always k-colours G. We prove that algorithm $B2()$ has polynomial average running time *for all sufficiently large values of n.*

Let τ be a *sufficiently small* constant. Divide the interval $[\epsilon,1]$ into subintervals $[\epsilon+l\tau, \min(1,\epsilon+(l+1)\tau)]$, $l \geqslant 0$. Now, using this division of $[\epsilon,1]$, divide the range $[n^{-1+\epsilon},1]$ into subranges $I_l = [n^{-1+\epsilon+l\tau}, \min(1,n^{-1+\epsilon+(l+1)\tau})]$, $l \geqslant 0$. Since ϵ and τ are constants, there is only a constant number of subranges I_l. Hence it is enough to prove that algorithm $B2(G,n,k)$ has polynomial running time for all sufficiently large values of n, as long as $p(n) \in I_l$ for some $l \geqslant 0$. Hence, in the rest of the proof we assume that $p(n) \in I_l$ for some $l \geqslant 0$. Let $p_{\text{low}} = n^{-1+\epsilon+l\tau}$ and $p_{\text{up}} = \min(1, n^{-1+\epsilon+(l+1)\tau})$. As before, we can assume that $p(n)$ is not in the last subrange.

Let $f(n) = n^{\gamma}$. Let $\delta = \min(\epsilon, -1+3\epsilon)$ be the constant mentioned in the proof of Theorem 4.3 and let $\delta_1 = \delta/2$. Since τ is sufficiently small, we have $\delta_1 > \tau$. First, we note that if $f(n) \geqslant (\log n)^3/p_{\text{low}}$, then we can apply Theorem 4.5 and deduce that for all sufficiently large values of n, $B2(G,n,k)$ has polynomial average running time. Hence, in the rest of the proof we assume that $f(n) \leqslant (\log n)^3/p_{\text{low}}$.

We use the same meanings (as used in the proof of Theorem 4.6) for $T_A(n)$, $T_i(n)$, $T_{bf}(n)$, $q(A), q(A,m)$. For the rest, their new meanings are given below.

- Let $m_1(n)$ be defined as $m_1(n) = \lceil f(n)/(n^{\tau} \cdot (\log n)^6)\rceil$. Let $n_f = n/(\log n)^3$. For $0 \leqslant j \leqslant r+1$, define $n_j = m_1(n) \cdot n^{\beta_j}$ where β_j are constants defined below. $E(T_{B2}(n))$ denotes the average running time of algorithm $B2$ over G.
- Define $\beta_0 = 0$. Let r be a nonnegative *integer constant* and let $\beta_1 \leqslant \ldots \leqslant \beta_{r+1}$ be a sequence of *positive real constants* such that

  (i) $0 < \beta_{j+1} - \beta_j \leqslant \delta_1$, for all $j \leqslant r$;

  (ii) $m_1(n) \cdot n^{\beta_r} \cdot (\log n)^3 \leqslant 1/p_{\text{up}}$ and $(1/p_{\text{low}}) \cdot (\log n)^3 \leqslant m_1(n) \cdot n^{\beta_{r+1}}$.

  Since δ_1 and τ are constants, and since τ is assumed to be sufficiently small, such constants r and $\{\beta_j\}$ always exist. The reason for choosing such constants is the following argument. FindColour2 (G,n,k,m) is useful for the analysis only when m is 'below' the barrier $1/p$. Also, Colour2(G,n,k,m,m) is useful for the analysis only when m is 'above' the barrier $1/p$.

From the earlier results we have the following.

By assumption, $q(A) \leqslant e^{-\Omega(f(n))}$.

- For all values of m such that $m \geqslant m_1(n)$ and $m \leqslant n_r$, FindColour2(G,n,k,m) succeeds on G with probability at least $1 - e^{-\Omega(m \cdot n^{\delta})}$. Hence, for $j \leqslant r$, we have $q(A, n_j+1) \leqslant e^{-n_j \cdot n^{\delta}}$.
- For all values of m such that $m \geqslant n_{r+1}$, Colour2(G,n,k,m,m) succeeds in k-colouring G with probability at least $1 - e^{-n \cdot k}$. Hence, for $j > r$, we have $q(A, n_j+1) \leqslant e^{-nk}$.

As before, we have

$$\begin{aligned} E(T_{B2}(n)) \leqslant\; & T_A(n) + O(n_0^2 \cdot n^{3 \cdot k \cdot n_0}) \cdot e^{-\Omega(f(n))} \\ & + \sum_{1 \leqslant j \leqslant r+1} O(n_j^2 \cdot n^{3 \cdot k \cdot n_j}) \cdot e^{-n_{j-1} \cdot n^{\delta}} + O(n^2 k^n) \cdot e^{-nk}. \end{aligned}$$

Substituting the values of n_j for various values of j, $1 \leqslant j \leqslant r+1$, we can verify that each of the last three terms on the right-hand side of the inequality is less than 1. Hence we have $E(T_{B2}(n)) \leqslant 2 \cdot T_A(n)$. Thus the algorithm $B2$ has polynomial expected running time. □

Corollary 4.1. *Let $G \in \mathcal{G}(n, p(n), k)$ where $p(n) \geqslant n^{-1+\epsilon}$ for some positive constant $\epsilon > 1/3$. Then G can be k-coloured in polynomial average time. (This follows from Theorem 3.3.)*

4.8. Third technique for designing polynomial average time algorithms

In this subsection, we describe the third technique for designing p.av.t. algorithms. It works by repeatedly calling Colour2, FindColour1, FindColour2 and FindColour3. Like the second technique, this technique also works for any exponentially low failure probability. However, this technique works for a larger range of edge probabilities, $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 1/4$.

Theorem 4.8. *Let $G \in \mathcal{G}(n, p(n), k)$, $p = p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 1/4$ be a random graph. Let A be a deterministic polynomial time (worst-case) algorithm such that A succeeds in k-colouring G with probability at least $1 - e^{-\Omega(n^{\gamma})}$, where γ is some positive constant. Then there exists a deterministic algorithm $B3(G, n, k)$ which always k-colours G and which runs in polynomial average time.*

Proof. The algorithm $B3(G, n, k)$ works as follows.

```
(1) Apply algorithm $A$ over the input random graph $G$; **if** $A$ succeeds, **then** exit.
(2) **for** $m = 1$ *to* $n/(\log n)^3$ **do**
        (2a) Apply Colour2$(G, n, k, m, m)$; **if** Colour2 succeeds, **then** exit.
        (2b) Apply FindColour1$(G, n, k, m)$; **if** FindColour1 succeeds, **then** exit.
        (2c) Apply FindColour2$(G, n, k, m)$; **if** FindColour2 succeeds, **then** exit.
        (2d) Apply FindColour3$(G, n, k, m)$; **if** FindColour3 succeeds, **then** exit.
(3) Apply brute-force colouring method over $G$.
```

It is clear that algorithm $B3$ always k-colours G. We prove that algorithm $B3(G, n, k)$ has polynomial average running time *for all sufficiently large values of n.*

First, we note that, if $\epsilon > 1/3$, then by Theorem 4.7 algorithm $B3(G, n, k)$ has polynomial average running time for all sufficiently large values of n. Hence, we assume that ϵ is such that $1/4 < \epsilon \leqslant 1/3$. Again, again by Theorem 4.7, we assume that $p = p(n)$ is such that $p(n) \geqslant n^{-1+\epsilon}$ and $p(n) \leqslant n^{-1+2/5}$. In the notation of Theorem 4.4, we have $\epsilon_1 = \epsilon$ and $\epsilon_2 = 2/5$.

Let τ be a *sufficiently small* constant. Divide the interval $[\epsilon, 2/5]$ into subintervals $[\epsilon + l\tau, \min(2/5, \epsilon + (l+1)\tau)]$, $l \geqslant 0$. Now, using this division of $[\epsilon, 2/5]$, divide the range $[n^{-1+\epsilon}, n^{-1+2/5}]$ into subranges $I_l = [n^{-1+\epsilon+l\tau}, \min(n^{-1+2/5}, n^{-1+\epsilon+(l+1)\tau})]$, $l \geqslant 0$. Since ϵ and τ are constants, there is only a constant number of subranges I_l. Hence it is enough to prove that algorithm $B3(G, n, k)$ has polynomial running time for all sufficiently large values of n, as long as $p(n) \in I_l$ for some $l \geqslant 0$. Hence, in the rest of the proof we assume that $p(n) \in I_l$ for some $l \geqslant 0$. Let $p_{\text{low}} = n^{-1+\epsilon+l\tau}$ and $p_{\text{up}} = \min(n^{-1+2/5}, n^{-1+\epsilon+(l+1)\tau})$. As before, we assume that $p(n)$ is not in the last subrange, without loss of generality.

Let $f(n) = n^{\gamma}$ and $g(n,p) = \max(1, 1/(np^2))$. Since $p(n) \leqslant n^{-1+2/5}$, we have $g(n,p) = 1/(np^2)$. Also, since $p \in I_l$, we have $g(n,p) \geqslant g_{\text{low}}(n,p)$ where $g_{\text{low}}(n,p) = n^{1-2\epsilon-2(l+1)\tau}$ and $g(n,p) \leqslant g_{\text{up}}(n,p)$ where $g_{\text{up}}(n,p) = n^{1-2\epsilon-2l\tau}$. Since τ is sufficiently small, we have $g_{\text{up}}(n,p) \cdot n^{4\tau} \cdot (\log n)^3 \leqslant 1/p_{\text{up}}$.

We note that if $f(n) \geqslant (\log n)^3/p_{\text{low}}$, then we can apply Theorem 4.5 and deduce that, for all sufficiently large values of n, algorithm $B3(G,n,k)$ has polynomial average running time. Hence, in the rest of the proof we assume that $f(n) \leqslant (\log n)^3/p_{\text{low}}$.

If $g_{\text{up}}(n,p) \cdot n^{\tau} \leqslant f(n)$, then it implies that the polynomial time algorithm A succeeds on G with probability at least $1 - e^{-\Omega(n^{\tau} \cdot \max(1,1/(np^2)))}$. Hence, by Theorem 4.6, algorithm $B3(G,n,k)$ runs in polynomial average time, for all sufficiently large values of n. Hence we assume that $g_{\text{up}}(n,p) \geqslant f(n)/n^{\tau}$ and hence $g_{\text{low}}(n,p) \geqslant f(n)/n^{3\tau}$. As a result, for any value of m such that $m \leqslant f(n)/(n^{3\tau} \cdot (\log n)^3) \leqslant g_{\text{low}}/(\log n)^3$, we have $m \cdot n \cdot p^2 \leqslant 1/(\log n)^3$.

Let δ_1 be a constant which is *less than but sufficiently close to* the constant $\min(\epsilon, -1+4\epsilon)$ used in the proof of Theorem 4.4. Since τ is sufficiently small, we have $\delta_1 \geqslant 6\tau$. Let δ_2 be a positive constant which is *less than but sufficiently close to* the constant $\min(4\tau, \epsilon)$ used in the proof of Theorem 4.2. Since τ is small, we have $\delta_2 \geqslant 3\tau$. Also, for future reference within this proof, we denote the constants $\min(\epsilon, -1+4\epsilon)$ and $\min(4\tau, \epsilon)$ by δ and δ', respectively.

We use the same meaning (as used in the proof of Theorem 4.6) for each of $T_A(n)$, $T_i(n)$, $T_{bf}(n)$ and $q(A), q(A,m)$. For the rest and some new notation, we use the following meanings.

- Let $m_1(n)$ be defined as $m_1(n) = \lceil f(n)/(n^{3\tau} \cdot (\log n)^3) \rceil$. Let $n_f = n/(\log n)^3$. For $0 \leqslant j \leqslant r+1+s+1$, define $n_j = m_1(n) \cdot n^{\beta_j}$ where β_j are constants defined below. $E(T_{B3}(n))$ denotes the average running time of algorithm $B3$ over G.
- Define $\beta_0 = 0$. Let r, s be nonnegative *integer constants* and $\beta_1 \leqslant \ldots \leqslant \beta_{r+1} \leqslant \ldots \leqslant \beta_{r+1+s+1}$ be a sequence of *positive real constants* such that

  (i) $0 < \beta_{j+1} - \beta_j \leqslant \delta_1$ for all $j \leqslant r$;

  (ii) $m_1(n) \cdot n^{\beta_r} \cdot (\log n)^3 \leqslant g_{\text{low}}(n,p)$ and $g_{\text{up}}(n,p) \leqslant m_1(n) \cdot n^{\beta_{r+1}}$;

  (iii) $0 < \beta_{j+1} - \beta_j \leqslant \delta_2$ for all $j \geqslant r+1$;

  (iv) $m_1(n) \cdot n^{\beta_{r+1+s}} \cdot (\log n)^3 \leqslant 1/p_{\text{up}}$ and $1/p_{\text{low}} \cdot (\log n)^3 \leqslant m_1(n) \cdot n^{\beta_{r+1+s+1}}$.

  Since δ_1, δ_2 and τ are constants, and since τ is assumed to be sufficiently small, such constants r, s and β_j always exist. The reason for choosing such constants is the following argument. FindColour3(G,n,k,m) is useful for the analysis only when m is 'smaller' than the 'barrier' $1/(np^2)$. Similarly, FindColour1 (G,n,k,m) is useful for the analysis only when m 'lies' between the two barriers $1/(np^2)$ and $1/p$. Also Colour2(G,n,k,m,m) is useful for the analysis only when m is 'above' the barrier $1/p$.

From the earlier results we have the following.

By assumption, $q(A) \leqslant e^{-\Omega(f(n))}$.

- For all values of m such that $m \geqslant m_1(n)$ and $m \leqslant n_r$, FindColour3(G,n,k,m) succeeds on G with probability at least $1 - e^{-\Omega(m \cdot n^{\delta})}$. Hence, for $j \leqslant r$, we have $q(A, n_j + 1) \leqslant e^{-n_j \cdot n^{\delta}}$.

- For all values of m such that $m \geqslant n_{r+1}$ and $m \leqslant n_{r+1+s}$, FindColour1(G, n, k, m) succeeds on G with probability at least $1 - e^{-\Omega(m \cdot n^{\delta'})}$. Hence, for $j > r$ and $j \leqslant r + 1 + s$, we have $q(A, n_j + 1) \leqslant e^{-n_j \cdot n^{\delta'}}$ where δ' was defined before as $\delta' = 4\tau$.
- For all values of m such that $m \geqslant n_{r+1+s+1}$, Colour2(G, n, k, m, m) succeeds in k-colouring G with probability at least $1 - e^{-n \cdot k}$. Hence, for $j > r + 1 + s$, we have $q(A, n_j + 1) \leqslant e^{-nk}$.

Hence we have

$$\begin{aligned} E(T_{B3}(n)) \quad &\leqslant \quad T_A(n) + \left[\sum_{1 \leqslant m \leqslant n_0} m \cdot T_m(n) \right] \cdot q(A) \\ &\quad + \sum_{1 \leqslant j \leqslant r+1} \left[\sum_{n_{j-1} < m \leqslant n_j} m \cdot T_m(n) \right] \cdot q(A, n_{j-1} + 1) \\ &\quad + \sum_{r+1 < j \leqslant r+1+s+1} \left[\sum_{n_{j-1} < m \leqslant n_j} m \cdot T_m(n) \right] \cdot q(A, n_{j-1} + 1) \\ &\quad + \left[\sum_{n_{r+1+s+1} < m \leqslant n_f} m \cdot T_m(n) + T_{bf}(n) \right] \cdot q(A, n_{r+1+s+1} + 1), \end{aligned}$$

$$\begin{aligned} E(T_{B3}(n)) \quad &\leqslant \quad T_A(n) + O(n_0^2 \cdot n^{3 \cdot k \cdot n_0}) \cdot e^{-\Omega(f(n))} \\ &\quad + \sum_{1 \leqslant j \leqslant r+1} O(n_j^2 \cdot n^{3 \cdot k \cdot n_j}) \cdot e^{-n_{j-1} \cdot n^{\delta}} \\ &\quad + \sum_{r+1 < j \leqslant r+1+s+1} O(n_j^2 \cdot n^{3 \cdot k \cdot n_j}) \cdot e^{-n_{j-1} \cdot n^{\delta'}} + O(n^2 k^n) \cdot e^{-nk}. \end{aligned}$$

Substituting the values of n_j for various values of j ($1 \leqslant j \leqslant r + 1 + s + 1$), we can verify that each of the last three terms on the right-hand side of the inequality is less than 1. Hence we have $E(T_{B3}(n)) \leqslant 2 \cdot T_A(n)$. Thus the algorithm $B3$ has polynomial expected running time. □

Theorem 4.9. *Let $G \in \mathscr{G}(n, p(n), k)$, $p(n) \geqslant n^{-1+\epsilon}$ where ϵ is a positive constant greater than 1/4. Then G can be k-coloured in polynomial average time.*

Proof. By Theorem 3.5, there exists a polynomial time algorithm A for k-colouring G such that A succeeds with probability at least $1 - e^{-\Omega(\min(np, n^3 p^4))}$ or with probability at least $1 - e^{-\Omega(\min(np, n^2 p^3))}$, depending on the value of $p(n)$. In either case, algorithm A succeeds on G with probability at least $1 - e^{-\Omega(n^{\gamma})}$ for some positive constant γ (since $\epsilon > 1/4$). Hence, by the previous theorem, there exists another algorithm $B3(G, n, k)$ such that $B3(G, n, k)$ always k-colours G and runs in polynomial average time. □

In each of Theorems 4.5, 4.6, 4.7 and 4.8, we require that A is a deterministic polynomial time (worst-case) algorithm that succeeds with probability $1 - f(n)$ for suitable $f(n)$. A close look at the proofs shows that we can weaken this requirement, thereby strengthening each of these theorems as follows. We state the strengthened version only for Theorem 4.5 (for others it is similar).

Theorem 4.10. *Let $G \in \mathscr{G}(n, p(n), k)$ where $p(n) \geqslant n^{-1+\epsilon}$ for some positive constant ϵ. Let A be any deterministic polynomial time (average case) algorithm that k-colours G with probability at least $1 - e^{-(\log n)^3/p(n)}$. Then we can construct another algorithm $B(G, n, k)$ which always k-colours G and whose running time is polynomial on average. (The difference between A and B is that B always k-colours, but A need not.)* □

5. Conclusions and open problems

We have obtained the following results. If $G \in \mathscr{G}(n, p(n), k)$, $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 1/4$, then

(i) G can be k-coloured in polynomial time with exponentially low failure probability;

(ii) G can be k-coloured in polynomial average time.

Even though our a.s. algorithms work only for $p(n) \geqslant n^{-1+\epsilon}$, $\epsilon > 1/4$, they have provably exponentially low failure probability, which is crucial in designing p.av.t. algorithms. We have shown how, by using the running time-failure probability trade-off, we can design p.av.t. algorithms. Our p.av.t. algorithms for k-colouring work for a much larger range of values of $p(n)$ than previously known algorithms. Both of our results can also be extended to the $\mathscr{GUC}(n, p(n), k)$ model for the same range of values of $p(n)$. The details are given in [15].

One natural question related to these results is as follows. Given a random k-colourable graph, can we find a $\chi(G)$-colouring in p.av.t.? With respect to worst-case measure, the two problems of k-colouring and $\chi(G)$-colouring are not equivalent. However, we have shown that $\chi(G)$-colouring can be done in p.av.t. for both random and semi-random k-colourable graphs for certain ranges of values of $p(n)$ and the results have been submitted. We suggest the following related open problems.

1. To design p.av.t. algorithms for k-colouring the $\mathscr{G}(n, p(n), k)$ and $\mathscr{GUC}(,,)$ models with $p(n) \geqslant n^{-1+\epsilon}$, where ϵ is any positive constant.

2. To design p.av.t. algorithms for $\chi(G)$-colouring random graphs from the $\mathscr{G}(n, p)$ model for constant p. Here every one of the $\binom{n}{2}$ edges is chosen with equal probability p. An easier problem is to design *approximation* algorithms with performance ratio bounded by a constant less than 2, and whose running time is polynomial on *average*. As mentioned in Section 1, the greedy colouring is an a.s. algorithm with a performance ratio of 2 [7].

Acknowledgement

The author thanks the anonymous referee for his feedback.

Note (added in proof)

The results on $\chi(G)$-colouring appear in *J. Algorithms* **33** (1999) 112–123. The results of this paper have been improved further and are going to appear in a future paper.

References

[1] Alon, N. and Kahale, N. (1994) A spectral technique for coloring random 3-colorable graphs. In *Proceedings of the* 26*th Annual ACM Symposium on Theory of Computing*, pp. 346–355.

[2] Blum, A. and Spencer, J. (1995) Coloring random and semi-random k-colorable graphs. *J. Algorithms* **19** 204–234.

[3] Dyer, M. E. and Frieze, A. M. (1989) The solution of some random NP-hard problems in polynomial expected time. *J. Algorithms* **10** 451–489.

[4] Feige, U. and Kilian, J. (1996) Zero knowledge and the chromatic number. In *Proceedings of the* 11*th Annual IEEE Conference on Computational Complexity*, pp. 24–27.

[5] Freize, A. M. and McDiarmid, C. (1997) Algorithmic theory of random graphs. *Random Structures and Algorithms* **10** 5–42.

[6] Garey, M. R. and Johnson, D. S. (1978) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

[7] Grimmett, G. R. and McDiarmid, C. J. H. (1985) On colouring random graphs. *Mathematical Proceedings of Cambridge Philosophical Society* **77** 313–324.

[8] Hagerup, T. and Rüb, C. (1989) A Guided Tour of Chernoff Bounds. *Information Processing Letters* **33** 305–308.

[9] Karger, D., Motwani, R. and Sudan, M. (1994) Approximate graph coloring by semi-definite programming. In *Proceedings of the* 35*th Annual IEEE Symposium on Foundations of Computer Science*, pp. 2–13.

[10] Karp, R. (1976) The probabilistic analysis of some combinatorial search algorithms. In *Algorithms and Complexity* (J. F. Traub, ed.), Academic Press, New York, pp. 1–19.

[11] Kučera, L. (1977) Expected behavior of graph colouring algorithms. In Vol. 56 of *Lecture Notes in Computer Science*, Springer, pp. 447–451.

[12] Shamir, E. and Upfal, E. (1984) Sequential and distributed graph coloring algorithms with performance analysis in random graph spaces. *J. Algorithms* **5** 488–501.

[13] Subramanian, C. R. (1994) Improved algorithms for coloring random graphs. In *Proceedings of the Fifth International Symposium on Algorithms and Computation*, Vol. 834 of *Lecture Notes in Computer Science*, Springer.

[14] Subramanian, C. R. (1995) Minimum coloring random and semi-random graphs in polynomial expected time. In *Proceedings of the* 36*th Annual IEEE Symposium on Foundations of Computer Science*, pp. 463–472.

[15] Subramanian, C. R. (1994) Algorithms for coloring random and semi-random graphs. PhD thesis, Department of Computer Science and Automation, Indian Institute of Science, Bangalore.

[16] Subramanian, C. R., Furer, M. and Veni Madhavan, C. E. (1998) Algorithms for coloring semi-random graphs. *Random Structures and Algorithms* **13** 125–158.

[17] Subramanian, C. R. and Veni Madhavan, C. E. (1994) Coloring semi-random graphs in polynomial expected time. In *Proceedings of the 14th Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, Vol. 880 of *Lecture Notes in Computer Science*, Springer, pp. 137–148.

[18] Turner, J. S. (1988) Almost all k-colorable graphs are easy to color. *J. Algorithms* **9** 63–82.