Smoothed Analysis for Graph Isomorphism

Michael Anastos
IST Austria
Klosterneuburg, Austria
Michael.Anastos@ist.ac.at

Matthew Kwan
IST Austria
Klosterneuburg, Austria
Matthew.Kwan@ist.ac.at

Benjamin Moore
IST Austria
Klosterneuburg, Austria
Benjamin.Moore@ist.ac.at

Abstract

There is no known polynomial-time algorithm for graph isomorphism testing, but elementary combinatorial "refinement" algorithms seem to be very efficient in practice. Some philosophical justification for this phenomenon is provided by a classical theorem of Babai, Erdős and Selkow: an extremely simple polynomial-time combinatorial algorithm (variously known as "naïve refinement", "naïve vertex classification", "colour refinement" or the "1-dimensional Weisfeiler–Leman algorithm") yields a so-called canonical labelling scheme for "almost all graphs". More precisely, for a typical outcome of a random graph $\mathbb{G}(n, 1/2)$, this simple combinatorial algorithm assigns labels to vertices in a way that easily permits isomorphism-testing against any other graph.

We improve the Babai–Erdős–Selkow theorem in two directions. First, we consider *randomly perturbed* graphs, in accordance with the *smoothed analysis* philosophy of Spielman and Teng: for any graph G, naïve refinement becomes effective after a tiny random perturbation to G (specifically, the addition and removal of $O(n \log n)$ random edges). Actually, with a twist on naïve refinement, we show that $O(n)$ random additions and removals suffice. These results significantly improve on previous work of Gaudio, Rácz and Sridhar, and are in certain senses best-possible.

Second, we complete a long line of research on canonical labelling and automorphisms for random graphs: for any p (possibly depending on n), we prove that a random graph $\mathbb{G}(n, p)$ can typically be canonically labelled in polynomial time. This is most interesting in the extremely sparse regime where p has order of magnitude c/n; denser regimes were previously handled by Bollobás, Czajka–Pandurangan, and Linial–Mosheiff. Our proof also provides a description of the automorphism group of a typical outcome of $\mathbb{G}(n, p)$ (slightly correcting a prediction of Linial–Mosheiff).

CCS Concepts

• **Theory of computation** → **Graph algorithms analysis**; • **Mathematics of computing** → **Combinatorics**; **Probability and statistics**.

Keywords

Graph Isomorphism, Smoothed Analysis, Weisfeiler-Leman Algorithm, Colour Refinement

ACM Reference Format:
Michael Anastos, Matthew Kwan, and Benjamin Moore. 2025. Smoothed Analysis for Graph Isomorphism. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC '25), June 23–27, 2025, Prague, Czechia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3717823.3718173

1 Introduction

Given a pair of graphs G_1 and G_2 (on the same vertex set $\{1, \dots, n\}$, say), how can we test whether they are isomorphic? Perhaps the most obvious first thought is to try to identify some easily computable isomorphism-invariant information that distinguishes the two graphs. For example, one can easily compute the degrees of the vertices of G_1 and the degrees of the vertices in G_2, sort these lists and check if they are the same. If they are different, we have successfully determined that the graphs are not isomorphic (and if they are the same our test was inconclusive).

Perhaps the most influential approach along these lines is called *colour refinement*, also known as *naïve refinement* or the *1-dimensional Weisfeiler–Leman algorithm*[1]. This is an algorithm that produces a "colour" for each vertex; the colour of a vertex describes the degree of that vertex, together with all other data that one can obtain by allowing degree information to "percolate through the graph". Briefly: at the start of the algorithm, every vertex "looks the same". Then, in the first step we distinguish vertices by their degrees (i.e., the colour of each vertex is its degree). In the second step, we distinguish vertices by their *number of neighbours with each degree* (i.e., each vertex now has a colour consisting of its own degree, together with a multiset of the degrees of its neighbours). In general, at each step we update the colour of each vertex by appending the multiset of colours of its neighbours. After some number of iterations of this procedure, it will "stabilise" in the sense that no further vertices can be distinguished from each other.

It turns out that the colour refinement algorithm can be executed very efficiently: in a graph with n vertices, the stable colouring can be computed in time [2] in time $O(n^2 \log n)$. It is not hard to see that *if* the algorithm manages to assign each vertex of G a *distinct label*, then the sequence of colours in the stable colouring uniquely determines the isomorphism class of G. In fact, more is true: in this case one can use the colours to define a *canonical labelling* of G. This notion is more formally defined in the full version of this paper, but roughly speaking it means that one can label the vertices

*All authors were supported by ERC Starting Grant "RANDSTRUCT" No. 101076777. Michael Anastos was also supported in part by the Austrian Science Fund (FWF) [10.55776/ESP3863424] and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101034413

STOC '25, Prague, Czechia

ACM ISBN 979-8-4007-1510-5/25/06
https://doi.org/10.1145/3717823.3718173

[1] The origin of this idea is difficult to pin down, but it seems to have been first proposed by Morgan [51] in 1965 in the context of computational chemistry!
[2] We are brushing over some subtleties here, which we discuss further in the full version of the paper.

of G with the integers $\{1, \dots, n\}$ (assuming G has n vertices), in such a way that isomorphic graphs are always labelled the same way.

Unfortunately, colour refinement is not always so effective. For example, in a regular graph, where every vertex has the same degree, colour refinement is useless on its own, as it cannot distinguish any vertices from each other. However, a landmark result of Babai, Erdős and Selkow [8] shows that this situation is "atypical": the proportion of n-vertex graphs which cannot be canonically labelled using colour refinement tends to zero as $n \to \infty$ (see also the improvements in [9, 40, 48]). This result is most easily stated in the language of random graphs, as follows.

Theorem 1.1. *For a random graph*[3] $G \sim \mathbb{G}(n, 1/2)$, *whp*[4] *the colour refinement algorithm distinguishes all vertices from each other. In particular, whp G can be tested for isomorphism with any other graph in polynomial time.*

Theorem 1.1 is widely touted as philosophical justification for why algorithms based on colour refinement seem to be so effective in practice. Indeed, all graph isomorphism algorithms in common usage employ a certain modification of colour refinement called *individualisation-refinement*[5]. It is well-known that algorithms of this type take exponential time on worst-case inputs (see [54] for recent general results in this direction), but one rarely seems to encounter such inputs in practice. Of course, we would be remiss not to mention that from a theoretical point of view, fully sub-exponential algorithms are now available: there is an amazing line of work due to Babai, Luks, Zemlyachenko and others (see [6] for a survey) applying deep ideas from group theory to the graph isomorphism problem, which culminated in Babai's recent quasipolynomial-time graph isomorphism [5] and canonical labelling [7] algorithms.

Remark. The study of the colour refinement algorithm is of interest beyond its direct utility in graph isomorphism testing. Indeed, if graphs G and H are indistinguishable by colour refinement, we say that G and H are *fractionally isomorphic*; this is an important notion of intrinsic interest in graph theory, that has surprisingly many equivalent formulations (e.g., in terms of first-order logic [21], universal covers [2], tree homomorphism counts [28] and doubly-stochastic similarity of adjacency matrices [59]). See for example the surveys [33, 55] and the monograph [13] for more.

We also remark that the colour refinement algorithm represents the limit of so-called *graph neural networks* for graph isomorphism testing [53, 62]; these connections have recently been of significant interest in the machine learning community (see [46, 52] for surveys).

[3]In the random graph $\mathbb{G}(n, p)$ (called the *binomial* or sometimes the *Erdős–Rényi* random graph), we fix a set of n vertices and include each of the $\binom{n}{2}$ possible edges with probability p independently.

[4]We say a property holds *with high probability*, or "whp" for short, if it holds with probability tending to 1. Here and for the rest of the paper, all asymptotics are as $n \to \infty$.

[5]It will not be relevant for the present paper to formally describe this paradigm, but for the curious reader: the idea is that occasionally a vertex must be artificially distinguished from the other vertices of its colour in order to "break regularity" (one must then consider all possible ways to make this artificial choice).

1.1 Smoothed Analysis

As our first main direction in this paper, we take the philosophy of Theorem 1.1 much further, combining it with the celebrated *smoothed analysis* framework of Spielman and Teng [58]. To give some context: the *simplex algorithm* for linear optimisation is another fundamental example of an algorithm which seems to perform well in practice but takes exponential time in the worst case. As a very strong explanation for this, Spielman and Teng proved that if one takes *any* linear optimisation problem and applies a slight (Gaussian) random perturbation to its coefficients, then for a typical outcome of the resulting perturbed linear optimisation problem, the simplex algorithm succeeds in polynomial time. This shows that poorly-performing instances are "fragile" or "unstable", and perhaps we should not expect them to appear in practice.

Linear optimisation is a continuous problem, and for discrete problems one needs a different notion of random perturbation. In a later paper, Spielman and Teng [57] suggested that the most natural way to define a discrete random perturbation is in terms of *symmetric difference*: for graphs G, G' on the same vertex set, write $G \triangle G'$ for the graph containing all edges which are in exactly one of G and G' (we can think of G' as the "perturbation graph", specifying where we should "flip" edges of G to non-edges, or vice versa). Significantly strengthening Theorem 1.1, we prove that for *any* graph, randomly perturbing each edge with probability about $\log n/n$ (i.e., adding and removing about $n \log n$ random edges[6]) is sufficient to make colour refinement succeed whp.

Theorem 1.2. *Fix a constant $\varepsilon > 0$ and consider any $p \in [0, 1/2]$ satisfying $p \geq (1+\varepsilon)\log n/n$. For any graph G_0, and with $G_{\text{rand}} \sim \mathbb{G}(n, p)$, whp the colour refinement algorithm distinguishes all vertices of $G_0 \triangle G_{\text{rand}}$ from each other. In particular, whp $G_0 \triangle G_{\text{rand}}$ can be tested for isomorphism with any other graph in polynomial time.*

Note that the $p = 1/2$ case of Theorem 1.2 is precisely Theorem 1.1. Indeed, when $p = 1/2$, the random perturbation is so extreme that all information from the original graph is lost and we end up with a purely random graph. (We may restrict our attention to $p \leq 1/2$, because perturbing G with a random graph $\mathbb{G}(n, p)$ is the same as perturbing the complement of G with a random graph $\mathbb{G}(n, 1-p)$.)

We remark that we are not the first to consider smoothed analysis for graph isomorphism: this setting was also recently considered by Gaudio, Rácz and Sridhar [31], though only under quite restrictive conditions on G_0, and with a stronger assumption on p (both of which make the problem substantially easier). Specifically, they found an efficient canonical labelling scheme that succeeds whp as long as G_0 satisfies a certain "sparse neighbourhoods" property, and as long as p has order of magnitude between $(\log n)^2/n$ and $(\log n)^{-3}$.

It is not hard to see that the assumption on p in Theorem 1.2 cannot be significantly improved. Indeed, if G_0 is the empty graph and $p \leq (1-\varepsilon)\log n/n$, then $G_0 \triangle G_{\text{rand}}$ is likely to have many isolated vertices (see e.g. [29, Theorem 3.1]), which cannot be distinguished by colour refinement. This does not necessarily mean that colour refinement fails to provide a canonical labelling scheme (isolated vertices can be labelled arbitrarily), but to illustrate a more serious

[6]Note that $\mathbb{G}(n, p)$ tends to have about $p\binom{n}{2} \approx pn^2$ edges.

problem, suppose G_0 is a disconnected graph in which every component is a 3-regular graph on at most 10 vertices. If we perturb with edge probability $p \leq (1/20)\log n/n$, it is not hard to see that whp many of the components of G will be completely untouched by the random perturbation (and therefore colour refinement will be unable to distinguish the vertices in these components, despite the components potentially having very different structure).

Of course, while tiny regular components may foil colour refinement, they are not really a fundamental problem for canonical labelling (as we can afford to use very inefficient canonical labelling algorithms on these tiny components). We are able to go beyond Theorem 1.2 via a modification of the colour refinement algorithm in this spirit. Indeed, with a variation on the colour refinement algorithm called the *2-dimensional Weisfeiler–Leman algorithm*, and a new notion of a *disparity graph* (which allows us to define an appropriate generalisation of the notion of a "tiny component"), together with an exponential-time canonical labelling algorithm of Corneil and Goldberg [22] (only applied to the analogues of "tiny components"), we are able to define a combinatorial canonical labelling scheme which becomes effective after extremely mild random perturbation (perturbation probability about $1/n$).

Theorem 1.3. *There is a set of graphs $\mathcal{H}$, and an explicit polynomial-time canonical labelling algorithm for graphs in $\mathcal{H}$ (which can also detect* whether *a graph lies in $\mathcal{H}$), such that the following holds.*

Consider any $p \in [0, 1/2]$ satisfying $p \geq 100/n$. For any graph G_0, and $G_{\text{rand}} \sim \mathbb{G}(n, p)$, whp $G_0 \triangle G_{\text{rand}} \in \mathcal{H}$. In particular, whp $G_0 \triangle G_{\text{rand}}$ can be tested for isomorphism with any other graph in polynomial time.

We highlight that the algorithm for Theorem 1.3 is a "combinatorial" algorithm, using completely elementary refinement/recursion techniques (in particular, the algorithm can be interpreted as falling into the *individualisation-refinement* paradigm, and it does not use any group theory). Actually, up to constant factors, the restriction $p \geq 100/n$ in Theorem 1.3 is essentially best possible for combinatorial algorithms, given state-of-the-art worst-case guarantees: if we were to take $G_{\text{rand}} \sim \mathbb{G}(n, p)$ for $p = o(1/n)$ (i.e., if we were to take a milder random perturbation than in Theorem 1.3), then any polynomial-time canonical labelling algorithm that succeeds whp for graphs of the form $G_0 \triangle G_{\text{rand}}$ would immediately give rise[7] to a sub-exponential-time canonical labelling algorithm for *all* graphs (i.e., an algorithm that runs in time $e^{o(n)}$). Although such algorithms are known to exist (most obviously, we already mentioned Babai's quasipolynomial time algorithm), they all fundamentally use group theory.

The proof of Theorem 1.3 involves a wide range of different ideas, which we outline at some length in Section 1.3. As an extremely brief summary: we first use expansion and anticoncentration estimates, together with a characterisation of colour refinement in terms of *universal covers*, to prove that the colour refinement algorithm (applied to $G_0 \triangle G_{\text{rand}}$) typically assigns distinct colours to the vertices in the so-called *3-core* of G_{rand}. This is already enough to prove Theorem 1.2, but to prove Theorem 1.3 (with its weaker assumption on p), we need to combine this with a delicate *sprinkling* argument, in which we slowly reveal the edges of G_{rand}, and study how the 3-core changes during this process. Every time a new vertex joins the 3-core (thereby receiving a unique colour by the colour refinement algorithm), this new colour information cascades through the 2-dimensional Weisfeiler–Leman algorithm, breaking up many of the colour classes into smaller parts. This has the effect of partitioning the graph into smaller and smaller parts, which can be treated separately at the end.

[7] For any graph G^*, we can consider a larger graph G_0 with many copies of G^* as connected components, with parameters chosen such that a very mild random perturbation typically leaves some copies of G^* completely unaffected (and therefore a canonical labelling of the randomly perturbed graph can be translated into a canonical labelling of G^*). We omit the details.

1.2 Sparse Random Graphs

After the Babai–Erdős–Selkow theorem (Theorem 1.1) on canonical labelling for $\mathbb{G}(n, 1/2)$, one of the most obvious directions for further study was to consider sparser random graphs $\mathbb{G}(n, p_n)$ (note that it suffices to consider $p_n \leq 1/2$, since canonical labelling is not really affected by complementation).

The first work in this direction was by Bollobás (see [17, Theorem 3.17]), who showed that the proof approach for Theorem 1.1 works as long as p_n does not decay too rapidly with n. Combining this with a later result of Bollobás [16] (which considered a very different type of canonical labelling scheme, under different assumptions on p_n), and a result of Czajka and Pandurangan [23] (which considered colour refinement for an intermediate range of p_n), one obtains polynomial-time canonical labelling schemes as long as $np_n - \log n \to \infty$ as $n \to \infty$. This condition on p_n is significant because it is the same range where random graphs are typically *rigid*: as was famously proved by Wright [61], such random graphs typically have only trivial automorphisms, whereas if $np_n - \log n \to -\infty$ then there typically exist many automorphisms.

Even sparser graphs were recently considered by Linial and Mosheiff [47]; they handled the regime where $np_n \to \infty$ using another different type of canonical labelling scheme. The regime $np_n \to 0$ is easy, as in this regime (see [29, Section 2.1]) the components of $\mathbb{G}(n, p_n)$ are all trees with size at most $o(\log n)$ (so various different types of trivial canonical labelling schemes suffice; see for example [4]). That is to say, the only regime left unaddressed is the regime where p_n is about c/n for some constant c.

As our next main result, we close this gap, finding a canonical labelling scheme for all p_n.

Theorem 1.4. *There is a set of graphs $\mathcal{H}$, and an explicit polynomial time canonical labelling algorithm for graphs in $\mathcal{H}$ (which can also detect* whether *a graph lies in $\mathcal{H}$), such that the following holds.*

For any sequence $(p_n)_{n\in\mathbb{N}} \in [0, 1]^{\mathbb{N}}$, and $G_n \sim \mathbb{G}(n, p_n)$, whp $G_n \in \mathcal{H}$. In particular, whp G_n can be tested for isomorphism with any other graph in polynomial time.

In fact, we prove that colour refinement on its own yields a canonical labelling scheme whp, unless p_n has order of magnitude c/n. If p_n has order of magnitude c/n, then colour refinement still *almost* works; the only obstruction is connected components which have a single cycle, which can be easily handled separately in a number of different ways.

Remark 1.5. There are important connections between the colour refinement algorithm and mathematical logic (see for example the monograph [32]). Our proof shows that random graphs of any

density typically have *Weisfeiler–Leman dimension* at most 2 (and if we delete components with a single cycle, the Weisfeiler–Leman dimension is exactly equal to 1). That is to say, random graphs of any density can typically be uniquely distinguished by the 2-variable fragment of first-order logic with counting quantifiers.

Remark 1.6. An equivalent version of Theorem 1.4, in the regime $p_n = O(1/n)$, has been independently proved in concurrent work by Oleg Verbitsky and Maksim Zhukovskii [60] (they also study the colour refinement algorithm, but with quite different methods: while we take advantage of our general machinery already used to prove Theorem 1.3, Verbitsky and Zhukovskii take advantage of structural descriptions of the "anatomy" of a sparse random graph [26, 27]). We believe both proof approaches to be of independent interest.

Given the machinery developed to prove Theorem 1.3, the proof of Theorem 1.4 is rather simple. Special care needs to be taken in the regime where p is very close to $1/n$ (this is the critical regime for the *phase transition* of Erdős–Rényi random graphs).

A related problem is to characterise the automorphism group of a random graph. Linial and Mosheiff achieved this when $np_n \to \infty$, and asked about the case where p_n has order of magnitude $1/n$ (specifically, they wrote that they "suspect" that the largest biconnected component of the so-called *2-core* has trivial automorphism group). We prove that Linial and Mosheiff's suspicion is mostly (but not exactly) correct. To state our theorem in this direction, we need a further definition.

Definition 1.7. For a graph G, let $\mathrm{core}_k(G)$ be its k-core (its largest subgraph with minimum degree at least k). A *bare path*in a graph is a path whose internal vertices have degree 2.

Note that any graph G can always be obtained by gluing a (possibly trivial) rooted tree to each vertex of $\mathrm{core}_2(G)$, and adding some tree components. As observed by Linial and Mosheiff, in order to characterise the automorphisms of G, it suffices to characterise how such automorphisms act on $\mathrm{core}_2(G)$. Indeed, having specified how an automorphism acts on $\mathrm{core}_2(G)$, all that remains is to specify automorphisms of the rooted trees attached with each vertex, and the tree components. (Note that automorphisms of rooted trees are easy to characterise; we can only permute vertices at the same depth, and only if their corresponding subtrees are isomorphic).

Theorem 1.8. *Consider any sequence $(p_n)_{n\in\mathbb{N}}$ and let $G \sim \mathbb{G}(n, p_n)$. Then, G satisfies the following property whp. Every automorphism of G fixes the vertices of the $\mathrm{core}_2(G)$, with the following (possible) exceptions.*

- *For every cycle component of $\mathrm{core}_2(G)$, automorphisms of this cycle may give rise to automorphisms of G.*
- *For every pair of cycle components in $\mathrm{core}_2(G)$ which have the same length, there may be an automorphism of G which exchanges these two cycles.*
- *For a pair of vertices u, v with degree at least 3 in $\mathrm{core}_2(G)$, such that (in $\mathrm{core}_2(G)$) there are multiple bare paths between u and v, there may be an automorphism of G which exchanges these bare paths.*
- *For a vertex u with degree at least 3 in $\mathrm{core}_2(G)$, such that (in $\mathrm{core}_2(G)$) there is a (closed) bare path from u to itself, there may be an automorphism of G which "flips" this bare path (reversing the order of the internal vertices).*

If $np_n \to \infty$, it is not hard to see that whp none of the above exceptions actually occur. Indeed, in the 2-core, whp: there is no pair of bare paths of the same length between any pair of degree-3 vertices, there are no bare paths from a vertex to itself, and there are no cycle components. However, when say $p_n = 2/n$, there is a non-negligible probability that each of the aforementioned configurations exist (and that the rooted trees attached to these configurations permit automorphisms of G which permute vertices of $\mathrm{core}_2(G)$): the asymptotic distribution of the numbers of each of these configurations can be described by a sequence of independent Poisson random variables with nonzero means. This can be shown by the method of moments (see for example [37, Section 6.1]); we omit the details.

Remark 1.9. In the independent work of Verbitsky and Zhukovskii mentioned in Remark 1.6, they also deduced Theorem 1.8. They also went on to characterise all the automorphisms of $\mathrm{core}_2(G)$ (not just those induced by automorphisms of G); this requires some additional work.

1.3 Key Proof Ideas

The details of the proofs of Theorems 1.2 to 1.4 and 1.8 can be found in the full version of this paper (which is available on the arXiv). In this extended abstract, we informally describe the key ideas in the proofs.

1.3.1 Exploring Universal Covers. The starting point for all the proofs in this paper is an observation essentially due to Angluin [2], that the stable colouring obtained by the colour refinement algorithm assigns two vertices the same colour if and only if the *universal covers* rooted at those vertices are isomorphic. We will use a minor modification of the notion of a universal cover called a *view*, which encodes the same information but which is slightly more convenient for our purposes. Roughly speaking, the view $\mathcal{T}_G(v)$ rooted at a vertex v in a graph G is a (potentially infinite) tree encoding all possible walks in G starting from v.

Without random perturbation, we are helpless to deal with the fact that many vertices may have isomorphic views (e.g., if G is d-regular, then $\mathcal{T}_G(v)$ is always isomorphic to the infinite rooted tree where each vertex has d children). The power of random perturbation is that, if two vertices u and v "see different vertices in their walks" (for example, if the set of neighbours of u is very different from the set of neighbours of v), then the random perturbation is likely to affect $\mathcal{T}_G(u)$ and $\mathcal{T}_G(v)$ differently, distinguishing them from each other. One of the key technical results in this paper is a general "exploration lemma" which makes this precise.

1.3.2 Distinguishing Vertices Via Random Perturbation: Expansion and Anticoncentration. It is difficult to give a quick summary of our exploration lemma without the necessary definitions, but to give a flavour: we define sets $\mathcal{S}^i(\{u, v\})$ which describe the vertices which "feature differently" in length-i walks starting from u and length-i walks starting from v. Our exploration lemma says that even extremely mildly randomly perturbed graphs typically have the property that if for some vertices u, v, and some i, the set $\mathcal{S}^i(\{u, v\})$

contains more than about $\log n$ vertices, then u and v are assigned different colours by colour refinement.

To prove this, we first use the expansion properties of random graphs (via a coupling of the type often appearing in analysis of branching processes) to "boost" the condition on $\mathcal{S}^i(\{u,v\})$, showing that if $\mathcal{S}^i(\{u,v\})$ has more than about $\log n$ vertices then there is typically some $j \geq i$ such that $\mathcal{S}^j(\{u,v\})$ has $n^{1-o(1)}$ vertices.

There is then a huge amount of "space" in $\mathcal{S}^j(\{u,v\})$ to take advantage of fluctuations due to the random perturbation. In particular, we show how to algorithmically divide $\mathcal{S}^j(\{u,v\})$ into "buckets" with different degree statistics, use the edges within these buckets to describe certain fluctuations in $\mathcal{T}_G(u)$ and $\mathcal{T}_G(v)$ via certain essentially independent inhomogeneous random walks on $\mathbb{Z}$, and then apply an anticoncentration inequality (a variant of the classical Erdős–Littlewood–Offord inequality) to show that it is extremely unlikely that these random walks behave in the same way for u and v (so unlikely that we can union bound over all u, v).

Of course, in order to actually apply our exploration lemma, we need to study the pairs u, v for which $\mathcal{S}^i(\{u,v\})$ has more than about $\log n$ vertices (for some i). If the perturbation probability is greater than about $\log n/n$ (as in Theorem 1.2), it is easy to show that whp *all* pairs of vertices have this property (in this case we can even take $i = 1$). For milder random perturbation (or sparser random graphs), we will need to restrict our attention to certain pairs u, v lying in certain special subgraphs, as we discuss next.

1.3.3 The 2-core and the Kernel. Recall from Definition 1.7 that the *k-core* of a graph G is its maximal subgraph of minimum degree at least k. The *kernel* of G is the smallest multigraph homeomorphic to the 2-core of G (i.e., with the same topological structure as the 2-core). It can be obtained from the 2-core by iteratively replacing bare paths (also defined in Definition 1.7) by single edges.

Especially in the setting of random graphs, cores and kernels are objects of fundamental interest, typically possessing extremely strong expansion properties. In particular, the kernel of a random graph in some sense describes its fundamental underlying expander structure: a celebrated theorem of Ding, Lubetzky and Peres [27] shows that one can in some sense "build a random graph from its kernel" by (very informally) starting from a random expander (the kernel), randomly replacing some edges with bare paths, (to obtain the 2-core), and randomly affixing some trees.

If $k \geq 3$, the expansion properties of the k-core make it quite convenient to apply our exploration lemma: it is fairly simple to show that, for $k \geq 3$, a mildly randomly perturbed graph typically has the property that colour refinement assigns a unique colour to all vertices of the k-core of G_{rand}. This immediately implies Theorem 1.2 (since when $p \geq (1+\varepsilon)\log n/n$ the 3-core of G_{rand} typically comprises the entire vertex set).

However, in very sparse regimes (in particular, when $p < c/n$ for a certain constant c, famously computed by Pittel, Spencer and Wormald [56]), the 3-core is typically *empty*, and we are forced to turn to the 2-core and the kernel. Unfortunately, this makes everything much more delicate. We prove a somewhat technical lemma showing that a mildly randomly perturbed graph typically has the property that colour refinement can distinguish the vertices[8] of degree at least 3 in the 2-core.

1.3.4 Sparse Random Graphs. The above considerations on the 2-core apply when $p \geq (1+\varepsilon)/n$ (for any constant $\varepsilon > 0$). Considering the case where the initial graph G_0 is empty, it is straightforward to deduce that simple canonical labelling schemes are typically effective for sparse random graphs $\mathbb{G}(n,p)$, when $p \geq (1+\varepsilon)/n$ (thus proving Theorem 1.4 in this regime).

The significance of this assumption on p is that it corresponds to the celebrated *phase transition* for Erdős–Rényi random graphs: when the edge probability is somewhat above $1/n$, there is typically a *giant component* with good expansion properties, but when the edge probability is somewhat less than $1/n$, there are typically only tiny components with very poor expansion properties (see for example the monographs [17, 29, 37] for more).

In the critical regime $(1-\varepsilon)/n < p < (1+\varepsilon)/n$, we proceed differently. In this regime, every component of $\mathbb{G}(n,p)$ with more than one cycle has quite large *diameter*: exploration processes can run for quite a long time without exhausting all the vertices in the graph, and we can accumulate quite a lot of independent randomness over this time. The anticoncentration from this randomness gives us another way to prove that different vertices are assigned different colours by colour refinement (completing the proof of Theorem 1.4).

1.3.5 Sprinkling Via the 3-core. For Theorem 1.3 (on canonical labelling of very mildly randomly perturbed graphs), the role of the k-core (and our lemmas about it) is that it provides a kind of "monotonicity" that allows us to use a technique called *sprinkling*.

For the unfamiliar reader: sprinkling, in its most basic form, is the observation that a random graph $G_{\text{rand}} \sim \mathbb{G}(n,p)$ can be interpreted as the union of two independent random graphs $G^1_{\text{rand}} \cup G^2_{\text{rand}}$, where $G^1_{\text{rand}}, G^2_{\text{rand}} \sim \mathbb{G}(n,p')$ with $1-p = (1-p')^2$. This observation allows one to first show that certain properties hold whp for G^1_{rand}, then reveal an outcome of G^1_{rand} satisfying these properties, and use the independent randomness of G^2_{rand} to "boost" these properties.

Sprinkling only really makes sense when we are dealing with properties of G_{rand} that are *monotone*, in the sense that once we have established the property for some subgraph of G_{rand}, adding the remaining edges of G_{rand} cannot destroy the property. Unfortunately, the colour refinement algorithm is highly non-monotone: in general, adding additional edges can allow the algorithm to distinguish more vertices, but can also prevent the algorithm from distinguishing some vertices. In the proof of Theorem 1.3, the critical role played by the ideas in Section 1.3.3 is that they makes a connection between the colour refinement algorithm and the k-core, which is a fundamentally monotone object. For example, if a vertex is in the k-core of G^1_{rand}, then it is guaranteed to be in the k-core of $G^1_{\text{rand}} \cup G^2_{\text{rand}}$.

We will actually split our random perturbation G_{rand} into many independent random perturbations $G^1_{\text{rand}}, \dots, G^T_{\text{rand}} \in \mathbb{G}(n,p')$ (in the proof of Theorem 1.3 we will take $T = 8$). If $p' \geq 10/n$, then one can show that whp the first random perturbation G^1_{rand} already has a

[8] This is not strictly speaking true; for the purposes of this outline we are ignoring a technical caveat concerning very rare configurations of edges in the 2-core.

giant 3-core[9], and (recalling Section 1.3.3) we can safely assume that each of the vertices in this 3-core will be assigned unique colours by the colour refinement algorithm (applied to the randomly perturbed graph $G = G_0 \triangle (G^1_{\text{rand}} \cup \cdots \cup G^T_{\text{rand}})$, which we have not yet fully revealed). Then, in each subsequent random perturbation G^i_{rand}, for $i \geq 2$, additional vertices randomly join the 3-core, and we can assume that they will also receive unique colours.

The upshot is that, whatever properties we are able to prove about the stable colouring produced by the colour refinement algorithm, we can "boost" these properties by randomly assigning unique colours to some vertices (and studying how this new information propagates through the colour refinement algorithm). To describe the types of properties we are interested in, we need to define the notion of a *disparity graph*[10], as follows.

1.3.6 Small Components in Disparity Graphs. Recall that Theorem 1.2 cannot hold for $p = o(\log n/n)$. The key obstruction to keep in mind is that if G_0 has many tiny regular connected components, then very mild random perturbation will leave some of these components untouched, and the colour refinement algorithm will not be able to distinguish the vertices in these components.

For the proof of Theorem 1.3, we therefore need an appropriate generalisation of the notion of "tiny component" (taking into account the fact that tiny components in the *complement* of G play the same role as tiny components of G). To this end, we introduce the notion of a *disparity graph*.

Definition 1.10. For a graph G, a set of colours Ω and a colouring $c : V(G) \to \Omega$, define the *majority graph* $M(G, c)$ (on the same vertex set as G) as follows. For any (possibly non-distinct) pair of colours $\omega, \omega' \in \Omega$:

- If at least half of the possible edges between vertices of colours ω and ω' are in fact present as edges of G, then $M(G, c)$ contains every possible edge between vertices of colours ω and ω'.
- Otherwise (if fewer than half of the possible edges between vertices of colours ω and ω' are present), $M(G, c)$ contains no edges between vertices of colours ω and ω'.

Then, define the *disparity graph* $D(G, c) = M(G, c) \triangle G$.

Informally speaking, the majority graph $M(G, c)$ is the best possible approximation to G among all graphs which are "homogeneous" between colour classes (for every pair of colour classes, to decide whether to put all edges between them or no edges between them, we look at the majority behaviour in G among vertices of those colours). Then, the disparity graph identifies the places where the majority graph differs from G. Equivalently, we can define the disparity graph to be the graph obtained by considering every pair of colour classes and deciding whether to complement the edges between those colour classes or not, depending on which of the two choices would make the graph sparser.

If c is the stable colouring obtained from the colour refinement algorithm, it is not hard to show that a canonical labelling of $D(G, c)$ yields a canonical labelling of G. So, we can canonically label G in polynomial time whenever $D(G, c)$ has sufficiently small components (small enough that we can afford to use known inefficient canonical labelling schemes on each component).

1.3.7 Percolation for a Weaker Result. Let c be the stable colouring obtained by the colour refinement algorithm. To prove Theorem 1.3, it would suffice to prove that whp the disparity graph $D(G_0 \triangle G_{\text{rand}}, c)$ has small components (for a polynomial-time combinatorial algorithm, we need every component to have $O(\log n)$ vertices, so that we can afford to use an exponential-time canonical labelling algorithm of Corneil and Goldberg [22] on each component[11]).

Unfortunately, we were not quite able to manage this when the random perturbation probability is $O(1/n)$ (as demanded by Theorem 1.3). Indeed, our proof of Theorem 1.3 requires a more sophisticated variant of the colour refinement algorithm, as we discuss later in this outline. However, the above goal is achievable if the random perturbation probability is at least (say) $100 \log\log n/n$. For expository purposes we next sketch how to prove this, before moving on to the more sophisticated ideas in the full proof of Theorem 1.3.

As discussed in Section 1.3.5, we can interpret our random perturbation G_{rand} as a composition of three random perturbations $G^1_{\text{rand}} \cup G^2_{\text{rand}} \cup G^3_{\text{rand}}$. The first random perturbation G^1_{rand} already whp establishes a giant 3-core (of size at least $n/2$, say); we can assume that the vertices in this 3-core get unique colours (and are hence isolated vertices in the disparity graph), so we only need to worry about the vertices outside the 3-core. Our two additional random perturbations G^2_{rand} and G^3_{rand} each cause an independent random subset of vertices to receive unique colours (as they join the 3-core): if $p \geq 100 \log\log n/n$, we calculate that each vertex receives a unique colour with probability at least $1 - o(1/\log n)$.

With one round of sprinkling (i.e., with the random assignment of unique colours provided by G^2_{rand}, followed by the colour refinement algorithm) we can show that the disparity graph has maximum degree $O(\log n)$ whp[12]. Indeed, the disparity graph describes how the neighbourhood of a vertex differs from the "majority behaviour" among vertices of its colour class, so if there is a vertex with high degree in the disparity graph, then there is a pair of vertices in the same colour class with very different neighbourhoods. With a union bound over pairs of vertices, it is easy to show that no such pairs persist after a round of sprinkling (the random assignment of unique colours typically allows the colour refinement algorithm to distinguish all such pairs).

For our second round of sprinkling (provided by G^3_{rand}), we view the random assignment of unique colours as *percolation*: if a vertex gets a unique colour, then it becomes isolated in the disparity graph, and we can imagine that that vertex is deleted. We are interested

[9] With a little more work, it would suffice to consider the vertices of degree at least 3 in the 2-core, in which case we only need p' to be slightly larger than $1/n$. However, the 3-core is much more convenient to work with.

[10] This notion has previously been introduced under the name "flip graph" [41, 42]; we thank the anonymous referees for bringing this to our attention.

[11] We remark that it does not actually seem to make the problem much easier if we weaken this requirement on the size of each component (which we may, if we are willing to use a more sophisticated group-theoretic algorithm, with better worst-case guarantees, on each connected component).

[12] For this bound we only need that (say) $p \geq 100/n$ (this implies that in each round of sprinkling, every vertex joins the 3-core with probability at least 0.9). Using that $p \geq 100 \log\log n/n$ (so in every round of sprinkling, every vertex joins the 3-core with probability at least $1 - o(1/\log n)$), we can actually show that the degrees are at most $O(\log n/\log\log n)$ whp. But, this stronger bound would not really affect the final result.

in the connected components that remain after these deletions[13]. But if the disparity graph has degree $O(\log n)$ before sprinkling, and each vertex is deleted with probability $1 - o(1/\log n)$, one can show that these deletions usually shatter the disparity graph into small components (the necessary analysis is similar to analysis of subcritical branching processes). This percolation step is the part where we are fundamentally using that the random perturbation probability is at least about $\log\log n/n$: if the perturbation probability were smaller than this, then the deletions would not be severe enough to break the disparity graph into small components.

1.3.8 Splitting Colour Classes and Components. In order to go beyond the ideas in the previous subsection, to work with random perturbation probabilities as small as $O(1/n)$, we need to get a much stronger conclusion from our sprinkled random perturbation. Every time a vertex is added to the 3-core (and gets a unique colour), this vertex is not simply removed from the disparity graph: the new colour information has a cascading effect (via the colour refinement algorithm) that affects the colours of many other vertices, indirectly affecting the connected components of the disparity graph.

First, it is instructive to think about the effect of sprinkling on the connected components (of the disparity graph) which intersect a given colour class C. If there is a vertex v which has a neighbour in C (with respect to the disparity graph), then assigning v a unique colour causes C to break into multiple colour classes: namely, the colour refinement algorithm will be able to distinguish the vertices in C adjacent to v from the vertices in C which are not adjacent to v (it is a simple consequence of the definition of the disparity graph that v is adjacent to at most half the vertices in C).

So, we can consider an exploration process that starts from the vertices of C, and explores[14] all the vertices which share a connected component with a vertex in C. Each time we consider a new vertex, we reveal whether it is assigned a unique colour, and propagate this information via the colour refinement algorithm[15]. As we continue this exploration/refinement process, the colour classes will start to break up into smaller pieces. There is a limit to how long this process can continue, since colour classes of size 1 cannot be broken up further. Indeed, by considering an auxiliary submartingale (which measures how the number of colours we have discovered so far compares to the number of vertices we have explored so far, at each point in the process), we can prove that our exploration process is likely to terminate after $O(|C|)$ steps, having explored all the vertices that share a component with a vertex in C. That is to say, the sizes of the connected components of the disparity graph (after sprinkling) are bounded in terms of the sizes of the colour classes (before sprinkling).

Unfortunately, we cannot hope to show that the colour classes are small, in general (indeed, if G has many isolated vertices, then all these vertices will be assigned the same colour by any canonical vertex-colouring scheme). However, the above idea can be "localised" to a connected component: We prove a crucial lemma, which tells us that in order to show that the disparity graph has connected components with $O(\log n)$ vertices (after sprinkling), it suffices to show that, before sprinkling, for all colour classes C and connected components X of the disparity graph, we have $|C \cap X| = O(\log n)$.

In order to show that these intersection sizes $|C \cap X|$ are small, we need another round of sprinkling which "shatters large components into small colour classes". In order for this sprinkling to have a strong enough effect, we need to consider a more powerful variant of the colour refinement algorithm, as follows.

1.3.9 Distinguishing Vertices Via the 2-dimensional Weisfeiler–Leman Algorithm. The *2-dimensional Weisfeiler–Leman algorithm* refines colourings of *pairs* of vertices: starting with a certain "trivial" colouring $\phi_G : V(G)^2 \to \Omega$, we repeatedly refine ϕ_G based on statistics of 3-vertex configurations, until a stable colouring $f : V(G)^2 \to \{0, 1\}$ is reached. This colouring of pairs of vertices then gives rise to a colouring of individual vertices $v \mapsto f(v, v)$, which contains a lot more information than the result of ordinary colour refinement.

In particular, this more sophisticated refinement operation allows us to distinguish vertices based on *distances*: in the final vertex-colouring, every two vertices of the same colour see the same number of vertices of every given colour at any given distance (in the disparity graph). As discussed in Section 1.3.7, after a single round of sprinkling whp the disparity graph has maximum degree $O(\log n)$, so if a connected component X has more than logarithmically many vertices then it is quite sparse, meaning that there is a very rich variety of different pairs of vertices at different distances. So, when sprinkling causes new vertices to receive unique colours, hopefully the 2-dimensional Weisfeiler–Leman algorithm will be able to significantly break up the colour classes in X (recall that our goal is now to prove that the intersections between colour classes and connected components have size $O(\log n)$).

Naïvely, one might hope to prove this via a simple union bound over subsets $Z \subseteq X$ of about $\log n$ vertices: for any such set Z, one might try to prove that after a round of sprinkling, and the 2-dimensional Weisfeiler–Leman algorithm, it is overwhelmingly unlikely that all vertices of Z have the same colour. To prove this, it would suffice to show that for any such Z there are many vertices which see some vertices of Z at different distances (so if any of these many vertices receive a new unique colour, this could be used by the 2-dimensional Weisfeiler–Leman algorithm to give different colours to some of the vertices of Z).

Unfortunately, this direct approach does not seem to yield any nontrivial bounds, without making structural assumptions about G. Instead, we use a "fingerprint" technique reminiscent of the method of *hypergraph containers* in extremal combinatorics, which reduces the scope of our union bound. Specifically, if there were a large colour class $C' \subseteq X$ after sprinkling, we show that this would imply the existence of a much smaller "fingerprint" set $S \subseteq X$ which sees many vertices (specifically, many vertices of C') at a variety of different distances. We can then take a cheaper union bound over the smaller fingerprint sets S.

We remark that the above sketch was very simplified, and serves only to illustrate the rough ideas. The full proof of Theorem 1.3 confronts a number of delicate technical issues and requires seven

[13] Here we are sweeping under the rug some technical issues related to the "consistency" of the disparity graph as the underlying graph changes. This turns out to be quite delicate!

[14] For the purposes of this outline, the reader can imagine that at each step we choose the unexplored vertex which is closest to C in the current disparity graph, though our actual exploration process is a bit more complicated.

[15] The resulting changes to the colouring also change the disparity graph. In particular, vertices which used to be in the same connected component may later be spread over multiple connected components, but this is not really a problem.

rounds of sprinkling, each of which gradually reduce the degrees, colour classes and connected components of the disparity graph.

1.4 Further Directions

There are a very large number of natural directions for further research. First is the question of optimising the probability implicit in the "whp" in each of Theorems 1.2 to 1.4, and improving the runtimes of the relevant algorithms. In the setting of Theorem 1.1, both these issues were comprehensively settled by Babai and Kucera [9]: they showed that, except with exponentially small probability, two steps of colour refinement suffice (yielding a linear-time algorithm).

The algorithm in Theorem 1.2 (colour refinement) runs in time $O((n+m)\log n)$, where n and m are the numbers of vertices and edges of our graph of interest, while the algorithm in Theorem 1.4 naïvely runs in time $O(n(n+m)\log n)$, due to repeated iteration of the colour refinement algorithm. With some careful analysis, it may be possible to bound the necessary number of colour refinement steps, to obtain an optimal linear-time algorithm in the setting of Theorem 1.2. In fact, the work of Gaudio, Rácz and Sridhar [31], mentioned earlier in the introduction, already makes some initial steps in this direction: they essentially show that three steps of colour refinement suffice, as long as p has order of magnitude between $(\log n)^2/n$ and $(\log n)^{-3}$. It seems plausible that in fact three steps of colour refinement suffice as long as $p \geq (1+\varepsilon)\log n/n$, and recent work on "shotgun reassembly" [30, 38] indicates that there should be a phase transition between two steps of colour refinement being necessary, and three steps being necessary, at around $p = (\log n)^2(\log\log n)^{-3}/n$.

With similar considerations on the necessary number of colour refinement steps, and the necessary number of iterations of the colour refinement algorithm, it may also be possible to improve the runtime in the setting of Theorem 1.4 to near-linear-time. Regarding Theorem 1.3: this algorithm also naïvely runs in about quadratic-time, due to the use of the 2-dimensional Weisfeiler–Leman algorithm. However, we do not need the full power of the general algorithms that we cite, and it seems plausible that special-purpose variants could be designed that might also yield a near-linear-time algorithm.

Also, there is the possibility of improving on the amount of random perturbation in Theorem 1.3. In particular, taking advantage of group-theoretic techniques (e.g., using the quasipolynomial-time algorithm of Babai [7]), it might be possible to design a canonical labelling scheme that becomes effective with tiny amounts of random perturbation (e.g., perturbation probability $o(1/n)$). In the case where G_0 is a regular graph, it may also be worth considering alternative models of random perturbation that do not destroy the regularity of G_0, as colour refinement is completely ineffective for regular graphs. Perhaps surprisingly, this may actually make the problem *easier*, as sparse random regular graphs are much better expanders than sparse Erdős–Rényi random graphs (cf. the work of Bollobás [16] showing that distance profiles provide an efficient canonical labelling scheme even for very sparse random regular graphs).

Next, there is the possibility of stronger connections between our results on smoothed analysis for graph isomorphism, and the work of Spielman and Teng on smoothed analysis for linear optimisation. Indeed, (a slight variant of) the colour refinement algorithm is used for *dimension reduction* in linear optimisation (see [34]) and it would be interesting to consider smoothed analysis in this setting.

There is also the possibility of considering smoothed analysis for many different types of graph algorithms other than isomorphism-testing. Some early work in this direction was undertaken by Spielman and Teng [57] (see also [11, 12, 49, 50]), but somewhat surprisingly, despite the algorithmic origin of the smoothed analysis framework, there is now a much larger body of work on randomly perturbed graphs in extremal and probabilistic graph theory (see for example [1, 3, 10, 14, 15, 18–20, 24, 25, 35, 36, 39, 43–45]) than on algorithmic questions.

In particular, it is worth remarking that canonical labelling is related to the so-called "network alignment problem" (also known as the "graph matching problem"), where the objective is to find a mapping between the vertex sets of two graphs such that the number of adjacency disagreements between the two graphs is minimised. It would be interesting to explore if this can be extended to the smoothed analysis setting.

References

[1] Elad Aigner-Horev, Dan Hefetz, and Michael Krivelevich. 2023. Cycle lengths in randomly perturbed graphs. *Random Structures Algorithms* 63, 4 (2023), 867–884. https://doi.org/10.1002/rsa.21170

[2] Dana Angluin. 1980. Local and global properties in networks of processors (Extended Abstract). In *Proceedings of the Twelfth annual ACM symposium on Theory of computing (STOC '80)*. ACM Press. https://doi.org/10.1145/800141.804655

[3] Sylwia Antoniuk, Andrzej Dudek, Christian Reiher, Andrzej Ruciński, and Mathias Schacht. 2021. High powers of Hamiltonian cycles in randomly augmented graphs. *J. Graph Theory* 98, 2 (2021), 255–284. https://doi.org/10.1002/jgt.22691

[4] V. Arvind, Johannes Köbler, Gaurav Rattan, and Oleg Verbitsky. 2017. Graph isomorphism, color refinement, and compactness. *Comput. Complexity* 26, 3 (2017), 627–685. https://doi.org/10.1007/s00037-016-0147-6

[5] László Babai. 2016. Graph isomorphism in quasipolynomial time [extended abstract]. In *STOC'16—Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, New York, 684–697. https://doi.org/10.1145/2897518.2897542

[6] László Babai. 2018. Groups, graphs, algorithms: the graph isomorphism problem. In *Proceedings of the International Congress of Mathematicians—Rio de Janeiro 2018. Vol. IV. Invited lectures*. World Sci. Publ., Hackensack, NJ, 3319–3336.

[7] László Babai. 2019. Canonical form for graphs in quasipolynomial time: preliminary report. In *STOC'19—Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. ACM, New York, 1237–1246. https://doi.org/10.1145/3313276.3316356

[8] László Babai, Paul Erdős, and Stanley M. Selkow. 1980. Random graph isomorphism. *SIAM J. Comput.* 9, 3 (1980), 628–635. https://doi.org/10.1137/0209047

[9] László Babai and Ludik Kucera. 1979. Canonical labelling of graphs in linear average time. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. IEEE. https://doi.org/10.1109/sfcs.1979.8

[10] József Balogh, Andrew Treglown, and Adam Zsolt Wagner. 2019. Tilings in randomly perturbed dense graphs. *Combin. Probab. Comput.* 28, 2 (2019), 159–176. https://doi.org/10.1017/S0963548318000366

[11] Nikhil Bansal, Daniel Reichman, and Seeun William Umboh. 2017. LP-based robust algorithms for noisy minor-free and bounded treewidth graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, Philadelphia, PA, 1964–1979. https://doi.org/10.1137/1.9781611974782.128

[12] Huck Bennett, Daniel Reichman, and Igor Shinkar. 2019. On percolation and $\mathcal{NP}$-hardness. *Random Structures Algorithms* 54, 2 (2019), 228–257. https://doi.org/10.1002/rsa.20772

[13] Christoph Berkholz, Paul Bonsma, and Martin Grohe. 2017. Tight lower and upper bounds for the complexity of canonical colour refinement. *Theory Comput. Syst.* 60, 4 (2017), 581–614. https://doi.org/10.1007/s00224-016-9686-0

[14] Tom Bohman, Alan Frieze, Michael Krivelevich, and Ryan Martin. 2004. Adding random edges to dense graphs. *Random Structures Algorithms* 24, 2 (2004), 105–117. https://doi.org/10.1002/rsa.10112

[15] Tom Bohman, Alan Frieze, and Ryan Martin. 2003. How many random edges make a dense graph Hamiltonian? *Random Structures Algorithms* 22, 1 (2003), 33–42. https://doi.org/10.1002/rsa.10070

[16] Béla Bollobás. 1982. *Distinguishing Vertices of Random Graphs.* Annals of Discrete Mathematics, Vol. 13. North-Holland, 33–49. https://doi.org/10.1016/s0304-0208(08)73545-x

[17] Béla Bollobás. 2001. *Random graphs* (second ed.). Cambridge Studies in Advanced Mathematics, Vol. 73. Cambridge University Press, Cambridge. xviii+498 pages. https://doi.org/10.1017/CBO9780511814068

[18] Julia Böttcher, Jie Han, Yoshiharu Kohayakawa, Richard Montgomery, Olaf Parczyk, and Yury Person. 2019. Universality for bounded degree spanning trees in randomly perturbed graphs. *Random Structures Algorithms* 55, 4 (2019), 854–864. https://doi.org/10.1002/rsa.20850

[19] Julia Böttcher, Richard Montgomery, Olaf Parczyk, and Yury Person. 2020. Embedding spanning bounded degree graphs in randomly perturbed graphs. *Mathematika* 66, 2 (2020), 422–447. https://doi.org/10.1112/mtk.12005

[20] Julia Böttcher, Olaf Parczyk, Amedeo Sgueglia, and Jozef Skokan. 2023. Triangles in randomly perturbed graphs. *Combin. Probab. Comput.* 32, 1 (2023), 91–121. https://doi.org/10.1017/s0963548322000153

[21] Jin-Yi Cai, Martin Fürer, and Neil Immerman. 1992. An optimal lower bound on the number of variables for graph identification. *Combinatorica* 12, 4 (1992), 389–410. https://doi.org/10.1007/BF01305232

[22] Derek Corneil and Mark Goldberg. 1984. A non-factorial algorithm for canonical numbering of a graph. *Journal of Algorithms* 5, 3 (Sept. 1984), 345–362. https://doi.org/10.1016/0196-6774(84)90015-4

[23] Tomek Czajka and Gopal Pandurangan. 2008. Improved random graph isomorphism. *J. Discrete Algorithms* 6, 1 (2008), 85–92. https://doi.org/10.1016/j.jda.2007.01.002

[24] Shagnik Das, Patrick Morris, and Andrew Treglown. 2020. Vertex Ramsey properties of randomly perturbed graphs. *Random Structures Algorithms* 57, 4 (2020), 983–1006. https://doi.org/10.1002/rsa.20971

[25] Shagnik Das and Andrew Treglown. 2020. Ramsey properties of randomly perturbed graphs: cliques and cycles. *Combin. Probab. Comput.* 29, 6 (2020), 830–867. https://doi.org/10.1017/s0963548320000231

[26] Jian Ding, Jeong Han Kim, Eyal Lubetzky, and Yuval Peres. 2011. Anatomy of a young giant component in the random graph. *Random Structures Algorithms* 39, 2 (2011), 139–178. https://doi.org/10.1002/rsa.20342

[27] Jian Ding, Eyal Lubetzky, and Yuval Peres. 2014. Anatomy of the giant component: the strictly supercritical regime. *European J. Combin.* 35 (2014), 155–168. https://doi.org/10.1016/j.ejc.2013.06.004

[28] Zdeněk Dvořák. 2010. On recognizing graphs by numbers of homomorphisms. *J. Graph Theory* 64, 4 (2010), 330–342. https://doi.org/10.1002/jgt.20461

[29] Alan Frieze and Michał Karoński. 2016. *Introduction to random graphs.* Cambridge University Press, Cambridge. xvii+464 pages. https://doi.org/10.1017/CBO9781316339831

[30] Julia Gaudio and Elchanan Mossel. 2022. Shotgun assembly of Erdős–Rényi random graphs. *Electron. Commun. Probab.* 27 (2022), Paper No. 5, 14. https://doi.org/10.1214/22-ecp445

[31] Julia Gaudio, Miklós Z. Rácz, and Anirudh Sridhar. [n. d.]. Average-case and smoothed analysis of graph isomorphism. *arXiv preprint arXiv:2211.16454* ([n. d.]). https://doi.org/10.48550/arXiv.2211.16454

[32] Martin Grohe. 2017. *Descriptive complexity, canonisation, and definable graph structure theory.* Lecture Notes in Logic, Vol. 47. Association for Symbolic Logic, Ithaca, NY; Cambridge University Press, Cambridge. ix+543 pages. https://doi.org/10.1017/9781139028868

[33] Martin Grohe, Kristian Kersting, Martin Mladenov, and Pascal Schweitzer. 2021. *Color Refinement and Its Applications.* The MIT Press, 349–372. https://doi.org/10.7551/mitpress/10548.003.0023

[34] Martin Grohe, Kristian Kersting, Martin Mladenov, and Erkal Selman. 2014. Dimension reduction via colour refinement. In *Algorithms—ESA 2014.* Lecture Notes in Comput. Sci., Vol. 8737. Springer, Heidelberg, 505–516. https://doi.org/10.1007/978-3-662-44777-2_42

[35] Max Hahn-Klimroth, Giulia S. Maesaka, Yannick Mogge, Samuel Mohr, and Olaf Parczyk. 2021. Random perturbation of sparse graphs. *Electron. J. Combin.* 28, 2 (2021), Paper No. 2.26, 12. https://doi.org/10.37236/9510

[36] Jie Han, Patrick Morris, and Andrew Treglown. 2021. Tilings in randomly perturbed graphs: bridging the gap between Hajnal–Szemerédi and Johansson–Kahn–Vu. *Random Structures Algorithms* 58, 3 (2021), 480–516. https://doi.org/10.1002/rsa.20981

[37] Svante Janson, Tomasz Łuczak, and Andrzej Rucinski. 2000. *Random graphs.* Wiley-Interscience, New York. xii+333 pages. https://doi.org/10.1002/9781118032718

[38] Tom Johnston, Gal Kronenberg, Alexander Roberts, and Alex Scott. [n. d.]. Shotgun assembly of random graphs. *arXiv preprint arXiv:2211.14218* ([n. d.]). https://doi.org/10.48550/arXiv.2211.14218

[39] Felix Joos and Jaehoon Kim. 2020. Spanning trees in randomly perturbed graphs. *Random Structures Algorithms* 56, 1 (2020), 169–219. https://doi.org/10.1002/rsa.20886

[40] Richard M. Karp. 1979. Probabilistic analysis of a canonical numbering algorithm for graphs. In *Relations between combinatorics and other parts of mathematics (Proc. Sympos. Pure Math., Ohio State Univ., Columbus, Ohio, 1978) (Proc. Sympos. Pure Math., XXXIV).* Amer. Math. Soc., Providence, RI, 365–378.

[41] Sandra Kiefer and Daniel Neuen. 2024. Bounding the Weisfeiler-Leman dimension via a depth analysis of I/R-trees. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science.* ACM, New York, 14. https://doi.org/10.1145/3661814.3662122

[42] Sandra Kiefer, Pascal Schweitzer, and Erkal Selman. 2021. Graphs Identified by Logics with Counting. *ACM Trans. Comput. Logic* 23, 1, Article 1 (oct 2021), 31 pages. https://doi.org/10.1145/3417515

[43] Michael Krivelevich, Matthew Kwan, and Benny Sudakov. 2017. Bounded-degree spanning trees in randomly perturbed graphs. *SIAM J. Discrete Math.* 31, 1 (2017), 155–171. https://doi.org/10.1137/15M1032910

[44] Michael Krivelevich, Daniel Reichman, and Wojciech Samotij. 2015. Smoothed analysis on connected graphs. *SIAM J. Discrete Math.* 29, 3 (2015), 1654–1669. https://doi.org/10.1137/151002496

[45] Michael Krivelevich, Benny Sudakov, and Prasad Tetali. 2006. On smoothed analysis in dense graphs and formulas. *Random Structures Algorithms* 29, 2 (2006), 180–193. https://doi.org/10.1002/rsa.20097

[46] Pan Li and Jure Leskovec. 2022. *The Expressive Power of Graph Neural Networks.* Springer Nature Singapore, 63–98. https://doi.org/10.1007/978-981-16-6054-2_5

[47] Nati Linial and Jonathan Mosheiff. 2017. On the rigidity of sparse random graphs. *J. Graph Theory* 85, 2 (2017), 466–480. https://doi.org/10.1002/jgt.22073

[48] Richard J Lipton. 1978. *The beacon set approach to graph isomorphism.* Technical Report 135, Yale University. Department of Computer Science.

[49] Avner Magen and Mohammad Moharrami. 2009. Robust algorithms for max independent set on minor-free graphs based on the Sherali-Adams hierarchy. In *Approximation, randomization, and combinatorial optimization.* Lecture Notes in Comput. Sci., Vol. 5687. Springer, Berlin, 258–271. https://doi.org/10.1007/978-3-642-03685-9_20

[50] Hermish Mehta and Daniel Reichman. 2022. Local treewidth of random and noisy graphs with applications to stopping contagion in networks. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques.* LIPIcs. Leibniz Int. Proc. Inform., Vol. 245. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, Art. No. 7, 17. https://doi.org/10.4230/lipics.approx/random.2022.7

[51] H. L. Morgan. 1965. The Generation of a Unique Machine Description for Chemical Structures—A Technique Developed at Chemical Abstracts Service. *Journal of Chemical Documentation* 5, 2 (May 1965), 107–113. https://doi.org/10.1021/c160017a018

[52] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. 2023. Weisfeiler and Leman go machine learning: the story so far. *J. Mach. Learn. Res.* 24 (2023), Paper No. [333], 59.

[53] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (July 2019), 4602–4609. https://doi.org/10.1609/aaai.v33i01.33014602

[54] Daniel Neuen and Pascal Schweitzer. 2018. An exponential lower bound for individualization-refinement algorithms for graph isomorphism. In *STOC'18—Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing.* ACM, New York, 138–150. https://doi.org/10.1145/3188745.3188900

[55] Oleg Pikhurko and Oleg Verbitsky. 2011. Logical complexity of graphs: a survey. In *Model theoretic methods in finite combinatorics.* Contemp. Math., Vol. 558. Amer. Math. Soc., Providence, RI, 129–179. https://doi.org/10.1090/conm/558/11050

[56] Boris Pittel, Joel Spencer, and Nicholas Wormald. 1996. Sudden emergence of a giant k-core in a random graph. *J. Combin. Theory Ser. B* 67, 1 (1996), 111–151. https://doi.org/10.1006/jctb.1996.0036

[57] Daniel A. Spielman and Shang-Hua Teng. 2003. Smoothed analysis: motivation and discrete models. In *Algorithms and data structures.* Lecture Notes in Comput. Sci., Vol. 2748. Springer, Berlin, 256–270. https://doi.org/10.1007/978-3-540-45078-8_23

[58] Daniel A. Spielman and Shang-Hua Teng. 2004. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. *J. ACM* 51, 3 (2004), 385–463. https://doi.org/10.1145/990308.990310

[59] Gottfried Tinhofer. 1986. Graph isomorphism and theorems of Birkhoff type. *Computing* 36, 4 (1986), 285–300. https://doi.org/10.1007/BF02240204

[60] Oleg Verbitsky and Maksim Zhukovskii. 2024. Canonical labelling of sparse random graphs. *arXiv preprint arXiv:2409.18109* (2024). https://doi.org/10.48550/arXiv.2409.18109

[61] E. M. Wright. 1970. Graphs on unlabelled nodes with a given number of edges. *Acta Math.* 126 (1970), 1–9. https://doi.org/10.1007/BF02392023

[62] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *International Conference on Learning Representations (ICLR) 2019* (2018).

Received 2024-10-29; accepted 2025-02-01