Combinatorics, Probability and Computing (2018) 27, 808–828.
doi:10.1017/S0963548318000123

An Average Case NP-complete Graph Colouring Problem†

LEONID A. LEVIN[1§] and RAMARATHNAM VENKATESAN[2]

[1] Department of Computer Science, Boston University, 111 Cummington Mall, Boston, MA 02215, USA
(home page: www.cs.bu.edu/fac/lnd/)

[2] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
(e-mail: venkie@microsoft.com)

Received 15 November 2015; revised 20 December 2017; first published online 2 April 2018

NP-complete problems should be hard on some instances but those may be extremely rare. On generic instances many such problems, especially related to random graphs, have been proved to be easy. We show the intractability of *random* instances of a graph colouring problem: this graph problem is hard on average unless all NP problems under all samplable (*i.e.* generatable in polynomial time) distributions are easy. Worst case reductions use special gadgets and typically map instances into a negligible fraction of possible outputs. Ours must output nearly random graphs and avoid any super-polynomial distortion of probabilities. This poses significant technical difficulties.

2010 *Mathematics subject classification*: Primary 60C05
Secondary 68Q17, 68Q25, 05C15, 05C80

1. Introduction

Many NP-complete problems are easy for random inputs: see *e.g.* [20]. Reductions $A \leqslant_f B$ between NP problems preserve only the worst case hardness: some instances of $B \supset f(A)$ are at least as hard as those of A. Johnson [23] and Gurevich [15] give a good account of some of the issues in average case analysis of NP problems. For instance, the Hamilton path problem takes linear average time [3, 13]. In such cases, NP-completeness may be misleading as evidence of hardness, *e.g.* for cryptography. (Many cryptography applications are based on pseudorandomness, pioneered in [8, 11, 33]; these in turn require hard on average one-way functions.) Such a disappointment has happened with the Knapsack problem [24, 30].

Another misinterpretation may cause premature abandonment of the search for algorithms efficient on all but extremely rare and peculiar instances. Some such algorithms are neat and

† An abstract of a 20-colour version of this result is in [31] and updates in arXiv:0112001.
§ This work was supported by NSF grant CCF-1049505.

simple, *e.g.* the graph isomorphism algorithm in [4]. Some problems (decoding of linear codes, versions of graph colouring and independent sets, *etc.*), however, have eluded such 'average case attacks'. So, one needs stronger hardness results related to 'typical' or 'average' instances of the problem. Such hardness would be sensitive to the choice of a particular NP-complete problem and its input distribution.

As evidence that a problem with a particular distribution is 'hard on average', we use a notion analogous to NP-completeness: our problem has no fast on average solution, *unless every NP problem under every samplable distribution has one*. In that case there is no chance of generating, in polynomial time, hard NP instances and the P =? NP question becomes academic. Other examples (tiling, matrix decomposition, *etc.*) are given in [14, 26]. These papers show that some NP problems with uniform probability distributions are *average case complete*. This concept was further analysed in [1, 6, 7, 18, 27, 28, 32, 34], and other works.

In typical NP-completeness proofs $A \leqslant_f B$, the reduction f of A to B uses special structures (gadgets) occurring only in a negligible fraction of instances of B except under strange distributions such as the one generated by f itself. Thus, the 'hard instances' of B may be extremely rare, and the problem may be easy on average under the distributions of interest. The need to avoid concentrating outputs to negligible sets presents a significant difficulty in designing f. Our f must output uniformly (within a polynomial factor) random graphs and only use gadgets available in them.

Below, we show the first such simple problem on graphs. This hardness result is to be contrasted with previous works that used random graphs only to point out that many NP-complete problems are easy on average. Moreover, as Gurevich [12] has shown, average case completeness of a *random graph* problem is *unlikely* (unless DEXP = NEXP) without *introducing randomizing reductions as we do here*.

To motivate our problem, we restate the classical edge-colouring problem (where all edges incident on a node must have distinct colours) in terms of 3-*graphs*, that is, 3-node induced subgraphs with induced colours and nodes relabelled 1, 2, 3: given a simple graph and m colours, edge-colour it so that no 3-graph contains two edges of the same colour. We generalize this notion by allowing the list C' of permissible types of 3-graphs to be arbitrary. As C' poses only local restrictions, we specify an additional global parameter: the number l of edges to be left blank. Set $C = (C', l)$. This formalism has notable power to express many graph problems, even using only one colour and blank. One example is the matching problem on a $2l$ node graph: C' contains 3-graphs with at most one blank edge each. Another is a similarly restated problem of finding an l-node clique in given graph: C' restricts blanks to be self-loops, all pairwise connected. However, our results below have no direct bearing on the complexity of these classical problems on random instances.

Our graphs are directed (digraphs). We will colour some of their edges with three colours, leaving l edges blank. So, each of the nine edges of a 3-graph has five options: to be coloured red, green, yellow, left blank, or be absent. C is taken by picking at random a number $l \in [1, n^2]$ and a subset C' of all 5^9 possible 3-graphs. A C-coloured graph is a graph so edge-coloured that it has l blank edges and all its 3-graphs are in C'. In the above examples a random C' is correct (for a given coloured graph) with a constant probability (if $m = O(1)$); l is with probability $1/n^2$.

Now, our problem admits a simple statement: given a random digraph G and a randomly chosen C, C-colour G. Using a randomized reduction we show the problem is complete in the

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

average case; thus, if this problem turns out to be easy on average, complexity-based cryptography will be impossible. Without requiring l blanks, the problem is trivial, but even with only one colour, *e.g.* red (and blanks), no polynomial-time algorithm is known: C can restrict all blank edges to be self-loops on a tournament.

We now compare our completeness with other proposed notions of average case intractability: they reduce the worst case instances of a problem A reputed to be hard to random instances of a problem B. Some such A are self-reducible (*i.e.* $A = B$). Or A is a variant of an NP-hard problem A' with altered parameters. In both cases A need not be NP-hard. We give some examples. Solving a noticeable fraction of instances would allow us to solve *all* instances for random self-reducible problems such as taking square roots modulo a composite, or discrete logarithm.[1] These examples randomize only arguments but not the modulus. Ajtai [1] randomizes the entire instance for a problem: given k, q and a random matrix A, solve $Ax \equiv 0 \pmod{q}, \|x\| = k$. This is shown to be at least as hard as the worst case versions of finding, up to a polynomial approximation factor, a shortest vector or a closest vector or a basis with the smallest orthogonality defect in lattices. The approximation factor is what separates these problems from NP-hardness, making such reductions (and analysis of lattice-based cryptosystems) possible.[2] The above examples may turn out to be easy on average, while some similar problems would still be hard. In contrast, our completeness ensures average hardness, unless no hard on average problems exist. The above examples have extra structure, attractive to cryptographic applications. Thus, if hard, they provide *one-way functions*, a very desirable tool similar to hard on average problems but not known to follow from the existence of the latter. And even from the existence of one-way functions, the hardness of the mentioned problems does not follow.

2. Basic concepts and claims

It is essential to consider reductions to (or hardness of) problems with *specific* distributions. If A is hard on average under some distribution, and $A \leqslant_f B$, then so is B under the induced distribution of f's outputs. But the latter may not correspond to any distribution of interest. It is interesting to show the reducibility under some common distributions, *e.g.* uniform distribution on graphs. To preserve their average case complexity, our reductions must map typical instances of one problem into typical instances of another. The definitions of distributions and averaging run times involve subtleties, some of which are discussed in this section.

Our strings and logarithms are binary, $\mathbb{S} = \{0,1\}^*$; $\|x\|$ is the bit-length of x; (x,y) is a string representing a pair of strings x, y. Algorithms' average run times may be considered on inputs with probability distributions on $\{0,1\}^n$, *e.g.* the family $\lambda_n(\{x\}) = 2^{-n}$. This may sometimes be artificial: the code lengths of two strings may be the same in some binary encodings of instances but differ in others. Instead, distributions may be over the entire $\mathbb{S}$, such as the uniform one $\lambda(\{x\}) = 2^{-n}/(n(n+1))$. But for tighter analysis on specific lengths, distributions μ_n may be conditioned to $\{0,1\}^n$. We say a property holds for *almost every* (a.e.) x if it fails with probability

[1] Over $GF(p)$ (find x from $(p, g \in \mathbb{Z}_p^*, g^x)$) or one elliptic curve or all [19] elliptic curves of the same order mod p. The best known algorithms (see [25]) run in conjectured time $\exp((\|p\|(\log\|p\|)^2)^{1/3})$.

[2] See *e.g.* [2]. Analogous rounding problems in non-Abelian discrete groups have tighter P-time approximation limits: any factor depending only on ambient dimensions would imply P = NP (see [5]).

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

$\mu_n = o(1)$. We may also consider uniform distribution on $\{0,1\}^{\mathbb{N}}$ and use a *cut-off* algorithm $C : \{0,1\}^{\mathbb{N}} \to \mathbb{S}$ selecting a finite prefix of a stream of bits after rejecting all shorter prefixes.

These probabilities differ by $\|x\|$ factors, but our completeness results ignore polynomial factors in probability and, thus, in average run time. This is similar to the worst case reduction theories, which typically ignore polynomial factors in run times. So, we use λ for any of these types of uniform distributions when unambiguous. Our distributions may sum to < 1 allowing processes a chance to fail in producing outputs. $\bar{R}(\mu)$ is the output distribution of R on μ-distributed inputs; $\bar{R}(\lambda)$ is *samplable* if R runs in polynomial time *(P-time)* with respect to output lengths. We use standard asymptotic notations $\Theta, O, o, \sim$.

2.1. Random Inversion Problems and reductions

Our version of NP is constructive: the witnesses must be explicitly produced. It is convenient to refer to them in the inversion form: for an algorithm f with $\|w\| = \|f(w)\|^{O(1)}$, given an *instance* x, find a *witness* $w \in f^{-1}(x)$. For example, $f((a,b))$ may output the graph a if b is its Hamiltonian cycle (or else an error). A *random inversion NP problem* (RIP) is a pair (μ, f) where f is computable in P-time and the distribution μ of the instances is samplable. The choice of a particular distribution μ is significant: different distributions may concentrate on different subsets of instances, thus specifying completely unrelated problems. RIPs require actually finding witnesses for instances, so our reductions are pairs (R, Q) of algorithms to map both. f_μ is $f(\mathbb{S})$ excluding x with $\mu(\{x\}) = 0$. R is (μ, f)-*injective* if $R(x) \neq R(y)$ for all $x \in f_\mu$, $y \in f(\mathbb{S}) \backslash \{x\}$.

Definition 2.1. A *reduction* of an RIP (μ, f) to (ν, g) is a pair (R, Q) of P-time[3] algorithms such that:

(1) R is (μ, f)-injective and $R(f_\mu) \subset g(\mathbb{S})$, *i.e.* R maps solvable instances to distinct solvable ones;
(2) $R(f(Q(w))) = g(w)$, *i.e.* Q solves f-instances x given a g-witness for $R(x)$;
(3) $\bar{R}(\mu)(\{y\}) \leqslant \nu(\{y\})\|y\|^{O(1)}$, *i.e.* R maps μ-typical instances into ν-typical ones.

The reductions are closed under composition and any fast on average algorithm for (ν, g) yields a fast on average algorithm for (μ, f). We consider two generalizations. A *padded* RIP (ν, g) is a modification of an RIP (ν, g') where $g((w, \alpha)) = (g'(w), \alpha)$. The distribution ν is only on $y = g'(w)$, combining all α. (Padding (ν, g) above with R's input circumvents the injection requirement.) A *randomization* (μ, f) of (μ', f'), is an RIP where the padding α is also random. It includes a 'cut-off' P-time algorithm $C(x, \alpha)$ that selects the padding length. The digits of α are chosen at random but α is restricted to sets s_x of measure $\lambda(s_x) = \|x\|^{-O(1)}$ (in our case actually ~ 1). Then $\mu(\{(x, \alpha)\})$ is $\mu'(\{x\})/2^{\|\alpha\|}$ for $\alpha \in s_x$, or 0 for $\alpha \notin s_x$.

Remark 2.2. s_x consists of those 'helpful' strings α which allow R to output instances y whose witnesses can be transformed by Q into witnesses for x, if any. s_x is not required to be decidable in

[3] The P-time requirement is essential for R only on average over x. This allows a possibility of generalizing completeness to problems, whose worst case versions are not NP-complete. Also μ may be not a samplable distribution – only dominated by one.

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

P-time, so this artificial restriction breaks samplability of μ. But s_x has a polynomial probability, so a randomized algorithm can try several α, shrinking exponentially the probability of failure for solvable instances. Some algorithms inverting g also give negative answers: 'no inverses exist'. Unlike inverses, they are not verifiable and so can be used in reductions only if their chance exceeds by $\|x\|^{-O(1)}$ a known $\lambda(s_x)$ bound.

Definition 2.3. A RIP (ν, g) is *complete* if every RIP has a randomization reducible to it (or to its padding).

The *average complexity* of (μ, f) is an upper bound $T(k)$ for the time needed to find $w \in f^{-1}(x)$ expressed in terms of $k(x) = \|x\| r(x)$, where the instance's 'rareness' or 'exceptionality' r has $\sum \mu(\{x\}) r(x) < \infty$. Reductions and randomizations preserve average complexity (up to a polynomial) and completeness. Note that complete problems *must* have many easy instances, since both hard and easy problems are reducible to them. Note also that if two distributions μ and μ' are such that $\mu(\{x\})/\mu'(\{x\}) = \|x\|^{O(1)}$, then $R(x) = x$ reduces (μ, f) to (μ', f): the polynomial factors in distributions are absorbed by the definition.

2.2. Main result

Let $\mathcal{G}$ be a digraph with edges (including *loops*, *i.e.* self-loops) coloured red, green or yellow, or left blank. Blank edges play a different role to coloured ones in that we limit the number of coloured (non-blank) edges.

A *spot* in $\mathcal{G}$ is a 3-node subgraph with induced edges and colours, and the nodes unlabelled (*i.e.* replaced by 1, 2, 3). So there are $< 5^9$ distinct spots. The *colouration* $C(\mathcal{G})$ is the set C' of all spots in $\mathcal{G}$ and the number l of its blank edges; $G = \overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by removing the colours. As noted in the Introduction, the classical edge-colouring problem restricts the set of spots so that all edges in a spot have different colours. We choose a random restriction instead. We similarly have 2-node spots called *links*.

Graph Colouration Problem. Invert the function $f(\mathcal{G}) = (\overline{\mathcal{G}}, C(\mathcal{G}))$, that is, colour the edges of a given graph to achieve the given colouration.

Uniform distribution. $\lambda(\{G\}) = 2^{-n^2}/(n(n+1))$, $\lambda(\{C'\}) = O(1)$, $\lambda(\{l\}) = n^{-2}$ for all C', l, and n-node graphs G.

Theorem 2.4. *Graph colouration is a complete Random Inversion Problem.*

We could also use a monotone variation, accepting any smaller C' and/or larger l. The function to invert would then map $(\mathcal{G}, C', l)$ to $(\overline{\mathcal{G}}, C', l)$, if $C(\mathcal{G}) = (\overline{C'}, \bar{l})$, $\overline{C'} \subset C'$, $\bar{l} \geqslant l$, to error otherwise.

2.3. Outline of the proof

The rest of the paper is devoted to proving this theorem. Section 3 has several lemmas on random graphs to show they are likely to have the structures (gadgets) needed for our reductions. (Some of these properties require tighter bounds than the literature offers so we must go into some computations.) One is the likely existence of a unique $(\log n)$-node tournament T. Edges between

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

a node and T define bits of a string we call node's *code*, used to order some nodes. Another property to mention is a bound on probability of existence of matchings that are used to embed specially designed large graphs of bounded degree into random graphs.

Section 4 defines a restricted type of tiling problem (RRTP) which retains its completeness and which we reduce to our graph colouring. Limited to 3-colour 3-node patterns, we must carefully select a universal Turing machine, split its symbols into bits/trits, and economically represent them in the tiling. We use a universal Turing machine from [17] and utilize its undefined transitions to implement our extras, *e.g.* non-deterministic choices.

A tiled square is encoded onto a coloured grid (represented as a graph) which we call a template. The grid squares have the graph's looped nodes at the corners and unlooped nodes in the centres. The edges connecting corners with adjacent corners and with the centres are to be coloured to reflect the tiling symbols. The reduction is based on embedding this template into a random graph. The grid nodes are those connected with all tournament nodes. The centres are placed in an order determined by their codes.

Section 5 presents our reduction. It generates roughly uniformly random graphs with an embedded instance of our Random Tiling Problem. We generate a graph at random, except for several features imposed on it. These features are likely to exist on their own, but may be hard to discover (*e.g.* it is not known how to find a $(\log n)$-node tournament in polynomial time). Some features are imposed for convenience, to reduce to $o(1)$ the probability of unsuitable graphs. Among these features are counts of certain types of nodes to match their expected value, the codes mentioned above being distinct, *etc.* One feature is of a different nature: the edges between successive grid centres must reflect the input bits of the tiling instance. This feature appears in random graphs only because the instance of tiling is itself randomly chosen.

Section 6 describes the required colouring patterns of all 3-node induced subgraphs and proves the equivalence of solving the tiling problem to the graph colouring it is reduced to. The tournament T is marked with blank loops; its two-way connections to looped nodes are allowed to include blank edges. The limit on the non-blank edges then ensures the $\log n$ size of T, which renders it unique. We then transform a solved tiling instance into a template coloured graph and use a lemma from Section 3 to embed this graph into the graph produced by the reduction. Any colouring of the graph to exhibit a solution to the GCP instance is forced to mark a grid on all nodes connected to the whole tournament. The colours of the grid edges must display the tiling solution. This used to be the trickiest part of the construction but is now much simpler.

3. Random graph lemmas

We describe the structures needed for the reductions and prove that they exist on a polynomial fraction of graphs. Such questions, when restricted to properties of almost all graphs, are addressed in the theory of random graphs (see [9]). Transitive tournaments (thereafter simply tournaments) are complete acyclic, except for loops at each node, digraphs. $\mathbb{E}_\mu(Y)$ stands for the expected value of the random variable Y with respect to the distribution μ. We omit μ when the distribution is understood. Unless stated otherwise, our random graphs have edges drawn independently with probability $1/2$. In this section we consider distributions on graphs with a given (not chosen at random) number of nodes; $f \sim g$ means $f(n) = (1 + o(1))g(n)$.

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

Remark 3.1. We use the following facts on probability and approximations.

1 (a) $n! = (n/e)^n\sqrt{2\pi n+\theta_n}$, where $\theta_n \in [\pi/3,\ e^2-2\pi] \subset [1,\ 1.11]$.

(b) For $k = o(\sqrt{n})$, $\binom{n}{k} \sim n^k/k!$.

(c) Let $k = o(\sqrt{n})$ integers be chosen from $\{1,\dots,n\}$ randomly and independently. The probability that all are distinct is $\Pi_{j=1}^{k-1}(1-j/n) = e^{-\Theta(k^2/n)} \sim 1$.

2 Let Y be a random variable taking each value $i \in \{1,\dots,s\}$ with probability $\mathbb{P}(\{i\}) = p_i > 0$. Let y_i be the number of occurrences of i in n independent Bernoulli trials of Y. Then, from the above item 1(a) and the central limit theorem:

(a) $np_i = \mathbb{E}(y_i)$; $\sigma_i^2 = \text{Variance}(y_i) = np_i(1-p_i)$.

(b) If $x^3 = O(\sigma_i)$, then $|y_i - np_i| \geqslant \sigma_i x$ with probability $e^{-x^2/2}/\Theta(x+1)$.

(c) Let k_i be integers with $\sum_i k_i = n$, let t_i be $(k_i/np_i)-1$, and let $\sum_i np_i t_i^2 = O(1)$. Then

$$\mathbb{P}(y_i = k_i,\ i = 1,\dots,s) = \sqrt{n}/O((2\pi n/s+1.11)^{s/2}).$$

Proof of item 2(c). The number of ways in which the event $[y_i = k_i, i := 1,\dots,s]$ can occur is $n!/\prod_{i\leqslant s}(k_i!)$ and each of these has probability $\prod_{i=1}^s p_i^{k_i}$. Thus the required probability is

$$n!\prod_{i=1}^s \frac{p_i^{k_i}}{k_i!} = n^n \prod_{i=1}^s \left(\frac{p_i}{k_i}\right)^{k_i} \sqrt{\frac{2\pi n+\theta_n}{\prod_{i=1}^s (2\pi k_i+\theta_{k_i})}}.$$

Now,

$$n^n \prod_{i=1}^s \left(\frac{p_i}{k_i}\right)^{k_i} = e^{-n(\sum_i p_i(1+t_i)\log(1+t_i))},$$

which we will now see is $1/O(1)$. Indeed, $\log(1+t) \leqslant t$, so

$$\sum_i np_i(1+t_i)\log(1+t_i) \leqslant \sum_i np_i t_i + \sum_i np_i t_i^2 = 0 + O(1),$$

by the assumptions. □

Lemma 3.2. *Let X_k be the number of k-node tournaments in a random n-node digraph, and let $c = k - \log n$, $|c| = o(k)$, $t = 2^{-kc}$. Then $\mathbb{E}(X_k) \sim t$; $\mathbb{E}(X_k^2) \sim t + t^2$.*

Proof. First, note that $X_k^2 = \sum_l N_l$, where N_l is the number of ordered pairs of k-node tournaments sharing $k-l$ nodes. By linearity of expectation, $\mathbb{E}(X_k^2) = \sum_l \mathbb{E}(N_l)$. There are $\binom{k}{l}$ ways to choose the positions of shared nodes in each tournament in a pair of k-node tournaments and $\binom{n}{k+l}(k+l)!$ ways to map this structure in our graph. The probability of each pair forming the tournaments is $2^{-(k^2-l^2+2kl)}$. Using item 1(b) of Remark 3.1, we get $\binom{n}{k+l}(k+l)! \sim n^{k+l}$, and substituting $n = 2^{k-c}$ below,

$$\mathbb{E}(N_l) \sim 2^{l^2-k^2-2kl}\binom{k}{l}^2 n^{k+l} = \binom{k}{l}^2 2^{-c(k+l)}2^{l(l-k)}.$$

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

Now, $\mathbb{E}(X_k) = \mathbb{E}(N_0) \sim t$, $\mathbb{E}(N_k) \sim t^2$, and

$$\mathbb{E}(X_k^2) = \sum_{l=0}^{k} \mathbb{E}(N_l) \sim t + t^2 + \sum_{l=1}^{k-1} \mathbb{E}(N_l).$$

It remains to show that $\sum_{l=1}^{k-1} \mathbb{E}(N_l) = o(t+t^2)$. Note that for $0 < l < k$, $2^{-c(k+l)} \leqslant 2^{-|c|}(t+t^2)$. So,

$$\mathbb{E}(N_l)/(t+t^2) \leqslant \binom{k}{l}^2 2^{l(l-k)} \leqslant \left\{ \begin{array}{l} 2^{-l((k-l)-2\log k)} \text{ for } 0 < l < k/2 \\ 2^{-(k-l)(l-2\log k)} \text{ for } k/2 \leqslant l < k \end{array} \right\} \leqslant 2k^2/2^k = o(1/k). \quad \square$$

Let $0 < c + 1/c = o(k)$ and let λ_c be the probability distribution of a $\lfloor 2^{k-c} \rfloor$-node graph G generated as follows. First we choose randomly a sequence of k nodes of G and force a tournament T on them. Then we choose at random the set of all other edges $(i,j) \notin V(T)^2$. The following corollary implies that a random graph on $\lfloor 2^{k-c} \rfloor$ nodes has a unique k-node tournament with a polynomial probability if and only if $c = O(1)$.

Corollary 3.3.

(1) *For those G with unique k-node tournaments, $\lambda_c(\{G\})/\lambda(\{G\}) \sim 2^{kc}$.*
(2) *λ_c-almost every G has only one k-node tournament.*

Proof. First note that λ_c-probability of all such pairs (T,G) is the same, and the total number of such pairs is $\sum_G X_k(G) = \mathbb{E}_\lambda(X_k(G))/\lambda(G)$. Thus, $\lambda_c(\{G\})/\lambda(\{G\}) = X_k/\mathbb{E}_\lambda(X_k)$, which is $\sim 2^{kc}$ if $X_k = 1$. Then the expected number of tournaments, other than the forced one, is $o(1)$:

$$\begin{aligned} \mathbb{E}_{\lambda_c}(X_k) &= \sum_G X_k(G)\lambda_c(\{G\}) \\ &= \sum_G X_k(G) \frac{X_k(G)\lambda(\{G\})}{\mathbb{E}_\lambda(X_k)} \\ &= \frac{\mathbb{E}_\lambda(X_k^2)}{\mathbb{E}_\lambda(X_k)} \sim \frac{2^{-kc} + 2^{-2kc}}{2^{-kc}} \\ &= 1 + 2^{-kc} \sim 1. \end{aligned}$$

$\square$

We use the next lemma on matchings in random graphs to show that a.e. G contains any $O(1)$-degree graph (*e.g.* Hamilton path, grids) as a spanning subgraph. Lemma 7.12 of [9] proves a.e. graph has perfect matchings. We need a tighter bound, so we use tighter calculations and a 'no solitary edge' argument.

Lemma 3.4. *Let G be a bipartite undirected graph with edges in $V_0 \times V_1$ chosen independently with probability b/n, $n = |V_i|$. Then, for large enough n, the probability σ that G has no perfect matching is $< 2n/e^b$.*

Proof. By Hall's theorem, if G has no perfect matching, it has an $(n+1)$-node independent set $L_0 \cup L_1$, $L_i \subset V_i$. Let L be a smallest such L_i in G. Then $a \stackrel{\text{def}}{=} |L| - 1 < n/2$, and no node has exactly

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

one edge to L (otherwise it can replace its neighbour, shrinking L). Let σ_a be the probability of G containing such an $L = L_0$. Then $\sigma < 2\sum_{a=0}^{(n-1)/2} \sigma_a$. Assume that $b < n < e^b/2$, otherwise the lemma is trivial. A node is isolated with probability

$$p = (1-b/n)^n < e^{-b-b^2/2n} < e^{-b}/(1+b^2/2n).$$

L_0, L_1 and two edges from each $x \in V_1 \backslash L_1$ to L_0 have

$$C_a = \binom{n}{a+1}\binom{n}{a}\binom{a+1}{2}^a$$

choices. For $a \in [2, n/2]$,

$$h \stackrel{\text{def}}{=} 1 - \frac{1}{a} - \frac{a+1}{n} \geqslant \frac{1}{2} - \frac{3}{n},$$

using

$$\binom{n}{k} < \left(\frac{ne}{k}\right)^k,$$

we have

$$\begin{aligned}
\sigma_a &< C_a \left(\frac{b}{n}\right)^{2a} \left(1-\frac{b}{n}\right)^{(n-a)(a+1)} \\
&< \left(\frac{ne}{a+1}\right)^{a+1} \left(\frac{ne}{a}\right)^a \left(\frac{a(a+1)}{2}\right)^a \left(\frac{b}{n}\right)^{2a} p^{ha+2} \\
&= \left(\frac{e^2 b^2 p^h}{2}\right)^a \frac{p^2 ne}{(a+1)} = o(p^2 n).
\end{aligned}$$

By inclusion–exclusion,

$$\sigma_0 < np - \binom{n}{2}p^2 + \binom{n}{3}p^3 < np\left(1 - p\frac{n}{3}\right).$$

Also,

$$\sigma_1 < \binom{n}{2} n \left(\frac{b}{n}\right)^2 p^{2-2/n} < \frac{n}{2} b^2 p^{2-2/n}.$$

So,

$$\sigma < 2\sigma_0 + 2\sigma_1 + o(p^2 n)n < 2ne^{-b}\left(1 - \frac{pn}{3} + \frac{b^2}{2}p^{1-2/n} + o(pn)\right) \Big/ \left(1 + \frac{b^2}{2n}\right) < 2ne^{-b}.$$

Indeed,

$$\frac{-pn}{3} + \frac{b^2}{2}p^{1-2/n} + o(pn) < b^2/2n,$$

or equivalently

$$\frac{b}{n} + \frac{n}{b}(2/3 - o(1))p > p^{1-2/n} b$$

by using $x + y \geqslant 2\sqrt{xy}$ and $b < |\log p|$. □

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

Lemma 3.5 (Equitable Colouring [16]). *Any undirected graph of degree $< d$ can be partitioned into d independent sets whose sizes differ at most by one.*

See [10] for a proof. Kierstead and Kostochka [22] give a P-time algorithm but we need only the existence of the partition. Next we use this lemma to find embeddings in random digraphs of some structures we need in our reduction. We first describe these embeddings.

Write $H \sqsubset G$ if H is an induced subgraph of G. We say that nodes u and v are *connected by a single edge* if exactly one of the two directed edges $(u,v),(v,u)$ exists, *connected by a double edge* if both do, and *non-adjacent* or *disconnected* if neither does. The degree $\Delta(u)$ of a node u is the number of adjacent nodes, and $\Delta(G)$ is the maximum degree of nodes in G. A *di-embedding* $g : V(G) \hookrightarrow V(H)$ is a bijection isomorphic on each *connected* 2-node subgraph of G. Note that disconnected nodes can be mapped to nodes that may be connected. We call two graphs *compatible* if they have the same number of looped and unlooped nodes.

Lemma 3.6 (Embedding Lemma). *Let $H \sqsubset F$, $d := \Delta(F)$, and $m := |V(F \setminus H)| = 4^d d^3/o(1)$. Choose digraphs G by dropping all non-loop edges incident on $F \setminus H$ and inserting randomly chosen edges instead. For a.e. such graph, there exists a di-embedding g of F into G which is an identity on H.*

We will use this lemma for a graph with $d = O(1)$: a grid with some additional edges. Now we want to find this grid in the random digraph by constructing a suitable g. We reduce the task of constructing g to that of finding perfect matchings in a sequence of random bipartite undirected graphs: using Lemma 3.5 we partition $F \setminus H$ suitably into some independent sets I_j and extend g on each I_j one by one.

Proof. *Initialization.* First we partition $F \setminus H$ into independent sets $I_j, j \leqslant d^2+1$, such that $|I_j| \geqslant m' := \lfloor m/d^2 \rfloor$, and each I_j consists of either only looped nodes or only unlooped nodes. Let F' be a graph obtained by adding edges between nodes in $F \setminus H$ that share a neighbour in H. Clearly $\Delta(F') < d^2$. Now, F' can be partitioned into independent sets using Lemma 3.5 as needed. Next, we split $V(G \setminus H)$ into disjoint V_j, compatible with $|I_j|$. (The latter condition can be met since G, F are compatible.) Set $K = H$. Below K will denote the set of nodes on which g is defined (*i.e.* di-embedding is found).

Repeat the following steps for $j := 1, 2, \ldots$, until the extension of g is completed.

Extension step (j). Call $v \in V_j$ a *candidate* for $u \in I_j$ if setting $g(u) := v$ will extend g as a di-embedding over $K \cup \{u\}$. For this, v must satisfy at most $2d$ connectivity constraints specifying which of the directed edges are present. Now we construct an undirected bipartite graph G'_j by connecting every node in I_j to all of its candidates in V_j. Then, with a high probability (over G), we find a perfect matching that maps I_j to V_j satisfying the connectivity requirements. This way we extend g over $K \cup I_j$ and update K to $K \cup I_j$.

Analysis of extension step (j). In G'_j the probability of an edge $\{u,v\}$ is $4^{-\Delta(u)}$. Also the edges are chosen independently: the conditional probability of v being a candidate for u is the same, given the set of all the edges from v to $\Gamma(I_j - \{u\}) \cap K$, since $\Gamma(I_j - \{u\})$ and $\Gamma(\{u\})$ have no

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

nodes in common in $G'_j \cup K$, which is guaranteed by construction of F' in the initialization and partitioning steps. The existence of matching is a monotone property, that is, it cannot be broken by adding edges. So, it suffices to estimate its probability by decreasing the chance of all edges in G'_j to the uniform 4^{-d}. Now we come to the overall success probability (over G) of finding g. Put $b/m' = 4^{-d}$. Then, using Lemma 3.4, the probability that one of the d^2 matching steps fails is

$$O(d^2 m/d^2)e^{-b} = O(m)/e^{m'/4^d} = O(e^{-(m/(d^2 4^d)-\log m)}) = o(1). \qquad \square$$

4. Turing machines and tilings

Our proof uses the completeness of the *Random Tiling Problem* (RTP). By [18], any samplable *Random Inversion Problem* (RIP) reduces to a problem with P-time computable distribution, *i.e.* with the measure of intervals $[0,x] \subset \mathbb{N}$ computable in time $\|x\|^{O(1)}$. In [26] Levin proved the RTP to be complete for such problems using deterministic reductions. A tile is a square with each corner labelled with a letter A–Z. A tiling of an $n \times n$ square involves covering it with n^2 tiles, so that the letters on the touching corners of adjacent tiles match. Our RIP will be inverting a function $\mathcal{T}$ that, given a tiled square input, returns the set of tiles (called *legal*) used in it, and its lowest (*floor*) row of tiles.

Problem. Invert $\mathcal{T}$, that is, given a set of legal tiles and a floor row, extend it into a tiled square.

Uniform distribution. choose n randomly with probability $1/(n(n+1))$, the set of legal tiles from all possible 26^4 tiles, and a legal tile at the lower left corner; choose each successive tile of the floor row with equal probability from the legal tiles matching those previously chosen. (If none exists, output a trivial instance.)

Our graphs are edge-coloured with only three colours, so we need careful restrictions on letters in the complete RTP we use. Figure 1 illustrates our RTP representing Turing machines (TMs). Our symbols have four fields: $\tau \in \{0,1,*\}$ (called a trit) and three bits b, s, s^-. We refer to s, s^- as the *direction* and *previous* direction (*leftward* $\Leftarrow$ or *rightward* $\Rightarrow$), respectively, to b as *priming* (and illustrate $b = 1$ by priming the trit, *e.g.* $* \mapsto *'$). No legal tile $\binom{xy}{zu}$ has $z_s \neq x_{s^-}$, or $u_s \neq y_{s^-}$, or $(x_s y_s) = (\Leftarrow\Rightarrow)$. Each row will consist of two segments: a rightward pointing segment at the left and a leftward pointing segment at the right. Tiles where z_s is $\Leftarrow$ have $y = u$. If u_s is $\Rightarrow$ then $x = z$. Two tiles cannot agree on both z, u but differ on both x, y. The left segment of the floor row must have $b = 1$, and the right segment must have $b = 0$, $\tau \neq *$. The top and floor symbols carry a special flag and must agree so that the entire tiled square can be wrapped into a cylinder. The cylinder has constant symbols at each end; it is further wrapped into a torus by a ring of the special *wall* tiles made by repeating this pair of symbols. We use just one particular tile set and it meets the above restrictions. Any tile set has a constant probability among all tile sets, so the density requirements of the reductions are not affected. Note that tiling an $n \times n$ square is easily reduced to tiling a larger $2p \times p$ rectangle with a prime $p \in [n, 2n]$. Our argument will use only such rectangles. We reduce this restricted form of the RTP (or *RRTP*) to the Graph Colouration Problem. Thus, we need to see that RRTP restrictions are compatible with reductions of [26] of non-deterministic computations by a universal TM to tiling.

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

(a) (b)

Figure 1. Space–time history for a right move $(\sigma_1, S_1) \to (\sigma_2, (S_2, RIGHT))$.

4.1. Completeness of the RRTP

Any RIP can be stated as accepting random instances by a run of a given non-deterministic P-time TM. All TMs can be simulated by a universal TM with a polynomial-time overhead and some special short (logarithmic) input prefix. We will now describe such a universal TM of [17].

Turing machines work on a tape consisting of a sequence of cells. Exactly one cell contains the TM's *head state S*; the others contain its tape symbols σ. A cell content at a given step is called an *event*. Their space–time table (*i.e.* a rectangular grid where *i*th row contains the entire tape contents at *i*th step) describes the history of the computation. The TM's head follows some path from its bottom row to the top. At each step the head acts on one of its two adjacent cells. We call this cell, the head's cell, and the border between them *active*, and the others *idle*. We now describe some *local* rules ensuring the global picture. The cells' content specifies the direction s to the active border. Thus, adjacent cells cannot both have s-directions facing away from each other. Idle cells do not change content from one row to the next. Active cells do, performing a *transition*. Exactly one of them changes direction and that one carries the head at the next step. Figure 1 pictures the space–time history around the active cells for the TM transition moving the head to the right. Its box containing the state S_1 points toward σ_1 (left or right as reflected in cases (a) and (b) respectively). In case (a), the right move merely causes the head to flip its direction; otherwise, the head switches places with the other active cell of the current row. Left moves work similarly as per the above rules.

A binary TM has $\{0,1\}$ tape alphabet. To convert a regular TM into a binary one M we must deal with its lack of a blank symbol which usually denotes the end of input. For this we prefix its input x with padding that encodes input length $l = \|x\|$ as a binary string preceded by a string of $2\|l\|$ zeros. In the cells carrying this padding, two counters are initiated that monitor the distance to both ends of the used part of M's tape (initially the input). M's head moving on the tape pulls these counters along and keeps them updated. When the right end of the used tape is reached, any subsequent characters are treated as blanks. M and its counter are initialized so that the head never approaches the tape ends. The simulated TM M may have a 'write 0 or 1' non-deterministic command. It is used to guess the solution of the instance. Our RRTP needs the space–time history to wrap around, making the top row consistent with the now adjacent bottom row. The non-deterministic command is used for this after a successful end to the computation.

We describe below a version of a TM U of [17] that simulates any such binary M. U has six leftward (*i.e.* with our $s = \Leftarrow$) and five rightward head states and six tape symbols. We represent them by the same fields τ, b (plus the directions s, s^-) as used in the RRTP. Head states differ

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

Table 1. Transition table

	$1'$	$0'$	$*'$	1	0	*
A				f	f	$e0$
B				F	F	$e1$
f,F			c	$b*$	$a*$	F
c	=	F	E'	$'$	$'$	
a	b'	F	E'	$'$	$'$	$'$
b		a'	D	$'$	$'$	$'$
d	$'$	$'$	D	$'$	$'$	
D			e'	d'	–	
E	$'$	$'$	e'	=	–	$'$
e	B	A	A/B	$'$	$'$	$'$

from tape symbols by $s \neq s^-$, *i.e.* changed from its previous value. The tape of U simulating M consists of a program segment P followed by the tape of the M segment denoted T. At each step, U marks the current cell it is working on, goes all the way left to the program segment, decides on the next step to perform, and returns to the current cell to do so. The tape trits in segment P never change. In segment T, the trits represent the current tape bits of M except that M's active tape symbol may be replaced by a $*$. This $*$ and the priming bits b are the only non-M data U uses. The simulation starts with P having only $0', 1', *'$ symbols followed by the head, then by $T \in \mathbb{S}$.

Table 1 has the $(*', e)$ command (one of the 'halt' commands in [17] denoted by $=$) modified to a choice of entering state A or B. This simulates M's non-deterministic command. We modify the initial state to be $(*', e)$, too; the effect of the non-determinism of U's first transition to A/B disappears in three steps – easy to see by tracing them. Afterwards, the simulation proceeds in a regular way.

The table's columns are indexed by tape symbols σ, the rows by the states (rightward directed, uppercase letters; leftward directed, lowercase letters). The entries reflect the resulting state (blank if unchanged), the trit of the new σ (blank if unchanged), and its bit (always, even if unchanged, *e.g.* in the transition from $(A, 1')$ the symbol $1'$ gets unprimed, while from $(a, 1')$ it stays primed).

The $(D, 0)$ command is unused. We employ it to mark a column of 'wall tiles' merging the left border of the space–time history with its right border to obtain a cylinder. U comes to its 'choose A or B state' command $(*', e)$ if the digit that M is to write in the Ikeno's representation of M's command is replaced with a $*$. Then U has a choice of acting as if it were 0 or 1.

5. The reduction algorithm *R*

The reduction R described below produces a graph G from any RRTP instance X supplemented with a random $O(\|X\|^7)$-bit α. R pairs G with a colouration: *standard* C' and l described in the next section. We show that C-colourings of such G exist for any solvable X, and all of them easily yield solutions for X.

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

Remark 5.1. Recall that the reduction (see Section 2.1) involves a pair of algorithms R and Q such that:

$$\text{(Randomized) tiling instance } Y=(X,\alpha) \xrightarrow{R} \text{graph colouration instance } (G,C).$$

$$\text{Solution } w' \text{ of tiling } Y \xleftarrow{Q} C\text{-coloured } G.$$

To ensure the solvability of $R(Y)$, Section 6 transforms a tiling solution to a coloured graph template G' and di-embeds G' into a suitably coloured G. This can be seen as a (not necessarily P-time) inverse of Q:

$$\text{Tiling solution } w' \xrightarrow{Q^{-1}} \text{edge-coloured graph } G' \hookrightarrow G.$$

Now we describe how $R(Y)$ designs the graph G. Let a prime p be the grid size of X, $k=\lceil \log_{1.5} 5p^2\rceil$, $n=\lfloor 4p^2(4/3)^k\rfloor$. First, R randomly selects disjoint sets T,U_T,L_T, in $|G|=\{1,\dots,n\}$ so that $|U_T|=|L_T|+1=2p^2$, $|T|=k$. Then R creates a graph G on $|G|=L\cup U$ (L being its set of looped nodes, U of unlooped nodes) randomly except that (i)–(iv) are enforced.

(i) Each node i has $(n/8)\pm x_i$, $x_i<n^{2/3}$ double edges to L. T forms a tournament $t_k,\dots,t_1$ (source to sink).

(ii) $L_T\subset L$, $U_T\subset U$. $U_T\cup L_T\cup T$ is the set of all nodes connected with every node in T.

The next condition uses the concept of *codes* defined in [4] as the $2|T|$-bit string, whose ith bit reflects the presence of the *code edge* $(u,t_{i/2})$ or, for odd i, $(t_{(i+1)/2},u)$. The *code* of u with respect to $t_i\in T$ is the restriction of its code to the digits $2i$ and $2i+1$.

(iii) All codes of U_T differ even in their $3/4$ most significant digits ($3/4$-*codes*).
Let $v_1,v_2,\dots v_{2p^2}$ be all the U_T nodes in decreasing order of codes and v_d be the first one disconnected from v_{d+1}.

(iv) t_1 has double edges to $v_1,v_p,v_{2p^2+1-p},v_{2p^2}$; the four connection types of v_i to v_{i+1}, $i\leqslant d$ reflect our RRTP floor's left (**s** is $\Rightarrow$) symbols τ; for $d<i<p$, the forward (v_{i-1},v_i) edges reflect its right τ bits.

Let $\mathcal{N}_2$ denote the set of graphs with unique k-node tournament, respecting (i), (ii); $\mathcal{N}$ also respects (iii), (iv).

Proposition 5.2. *In a.e. G all nodes in U_T have distinct $3/4$-codes.* □

Proof. The codes of nodes in U_T are independent and uniformly distributed for graphs in $\mathcal{N}_2$. There are $x=3^{3k/4}$ possible $3/4$-codes for U_T, and $|U_T|<y=(3/2)^k$. The chance that all codes are distinct is $(1-1/x)^{y(y-1)/2}>e^{-y(y-1)/(2x-2)}>e^{-y^2/x}\sim 1$, since $y^2/x=((3/2)^2/3^{3/4})^k=(243/256)^{k/4}=o(1)$. □

Remark 5.3. Almost every graph generated by R has a unique k-tournament, unique codes for U_T, and so it is in $\mathcal{N}$ and uniformly distributed on it. Thus, we will consider the uniform distribution on $\mathcal{N}$ and denote it by $\mathbb{P}$. We define s_X in Section 2.1 to contain exactly those α in Y resulting in graphs $G\in\mathcal{N}$ that satisfy all properties of (a.e.) G required below. Thus, these properties of G will be assumed.

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

The theorem follows from the two following propositions. The next proposition ensures the gadgets produced by R and used in our reduction are available in uniformly random graphs.

Proposition 5.4. *The probabilities of graphs G output by R (from λ-random X, α) then are $O(\|\alpha\|\lambda(\{G\}))$.* □

Proof. Assigning edges between successive nodes in U_T to encode a *random* RRTP instance agrees with λ. Thus we only need to show that other constraints R enforces are satisfied by a sufficient fraction of graphs.

By Corollary 3.3 and Proposition 5.2, the set H of graphs with a k-node tournament T have probability $\lambda_c(H) \sim 2^{kc}\lambda(H) = \lambda(H) \cdot O(n)$. With such T, the probability of sizes of L_T and U_T being their expected value $2p^2$ is $1/O(n)$ by item 2(c) of Remark 3.1. All other conditional probabilities are $1 - o(1)$, except for four t_1-links being double, which is $1/81$. □

Proposition 5.5. *With all X and a.e. α, the reduction R succeeds on $Y = (X, \alpha)$, that is, $G = R(Y)$ has a C-colouring w if and only if X has a tiling w'. Moreover, w is constructed in P-time, by $Q(w)$.* □

The proof is the topic of the next section. It describes the standard colouration C required by R and the inverse Q^{-1} of Q mapping tiling solutions to C-coloured graphs. The crucial part is to ensure item (2) of Definition 2.1: that Q can transform any witness of colouring problem into a witness of tiling.

6. The colouration C and the proof of Proposition 5.5

Enforcing structures with spots: bootstrapping. We now give simple and typical examples of how our design of C is used, restricting spots allowed in the graph (called C-spots). Properties enjoyed by all C-coloured graphs are said to be *forced by* C. For instance, our C has only one spot with three blank (blank-looped) nodes and it is a tournament. This forces all blank nodes of any C-coloured graph to form a tournament as well.

Another example: no C-spots have nodes with two links of certain types, *e.g.* outgoing red edges without reverse. This imposes the same behaviour on any C-coloured graph. Yet, local conditions alone, such as those imposed by spots, cannot ensure all needed features, *e.g.* at least one link of a given type at certain nodes. But our C also requires us to leave enough edges blank. We show this ensures the maximal size of the above tournament T (and thus its unique location, bootstrapping our construction). Indeed, some links with blank edges are only at blank nodes, and any smaller T lowers the number of blank edges obtainable by using only C-spots.

Next we describe a coloured graph template reflecting any solved tiling instance and random graphs matching its input row. We force it to be displayed by any adversary C-colouring such a random graph.

Mapping the tiling to a coloured template. We first make the loops of T blank and make all double (*i.e.* double-edge) links between L and T blank-yellow. We then implement Q^{-1} as outlined in Remark 5.1. First, we transform the tiling into a coloured 'template' graph G'. Then

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

we di-embed $G' \setminus G$ into G, copying the colours. All other edges in G will be yellow. C will ensure any colouring of G has this form. See Figure 2.

Our G' consists of T, U_T and, in addition, a toroidal *grid*: a product graph $\mathbb{Z}/2p \times \mathbb{Z}/p$, including $z_0 = t_1$ located at the *origin* of the grid. A grid's smallest *squares* (undirected 4-node cycles) represent RRTP tiles and are ordered by rows and by columns in each row. The corners of each ith square are connected by *radial* links into its *centre* $v_i \in U_T$ (treated thereafter as part of the grid). We also need the random edges of the graph G to determine uniquely the colouring of the grid's *floor row* containing z_1. For that we include in G' its code links to T. They ensure (Section 6.2) the correct placement of v_i centres in the first row.

(1) The codes of Section 5 determine a monotone order of v_i nodes forming the *input chain*.
(2) The colouring of the code edges proves the correctness of this ordering.
(3) The connections of the input chain consecutive nodes determine the colouring of the grid's floor row.
(4) The colouring of the rest of the grid exhibits a solution to the tiling problem instance.

We may denote colours as R (red), G (green), Y (yellow) and loosely refer to the absence or blankness of an edge as two special 'colours' (A and B). An edge-coloured red (resp. green, *etc.*, or left blank) is called a red edge (resp. green, blank, *etc.*), and the same for double edges, for example, a yellow edge with reverse blank edge is yellow-blank. The node's colour (or blankness) is that of its loop. We denote the induced coloured 3-node and 2-node subgraphs (spots and links) as $[u, v, w]$ and $[v, t]$, respectively. Our grid's L-nodes have two vertical $\mathbf{v}, \mathbf{v}^{-1}$, two horizontal $\mathbf{h}$ and four radial $\mathbf{i}, \mathbf{j}, \mathbf{k}, \mathbf{l}$-links each. The $\mathbf{h}$-links are directed *idle* or undirected: *active* and *wall*. We arbitrarily refer to active links as *outgoing* and wall links as *incoming* at both ends.

With these conventions each grid node has exactly one link of each type $\mathbf{i}, \mathbf{j}, \mathbf{k}, \mathbf{l}$, and L-nodes have one incoming and one outgoing $\mathbf{h}$ and $\mathbf{v}$. Thus we can treat them as functions: if $\mathbf{v}$ connects x to y and $\mathbf{i}$ connects y to z, we write $y = \mathbf{v}(x)$, $z = \mathbf{i}(y) = \mathbf{iv}(x)$. Except for non-idle $\mathbf{h}$, we can also use inverses, $x = \mathbf{v}^{-1}(y)$. We define $\mathbf{h}'$ as $j.i$ and will prove this permutation commutes with $\mathbf{v}$. The column of the origin z_1, we call the *wall*.

We use $\mathbf{I}$-links to denote $\mathbf{i}$ in L_T, $\mathbf{k}$ from U_T to the wall, or $\mathbf{j}$ at the rest of U_T. They link our input chain $z_1, v_1, z_2, v_2, z_3, \ldots, z_{2p^2}, v_{2p^2}, z_i \neq z_j$ if $i \neq j$. It passes through all $2p$ floor nodes, descends to the wall of the next lower rows via $\mathbf{k}$, and eventually passes through all the grid nodes.

The colouring of $\mathbf{v}$-links represents a solution to the tiling problem. This includes a direction bit $\mathbf{s}$ (also in $\mathbf{i}, \mathbf{l}$) used below; $\mathbf{j}$ at z_1 carries wall $\mathbf{w}$ and floor $\mathbf{f}$ flags, and they propagate: $\mathbf{w}$ via $\mathbf{j}, \mathbf{l}$ chain, $\mathbf{f}$ via $\mathbf{j}, \mathbf{i}$ chain. More details can be found in Section 6.3. Then we use the key fact that $G' \setminus T$ *has degree* $O(1)$. In Section 6.3 this allows us to prove the existence of an identical on $T \cup U_T$ di-embedding g_x of G' into a.e. G.

6.1. Forcing the blank tournament

This section shows how a selection of spots with blank edges enforces a $\mathbb{Z}/2p \times \mathbb{Z}/p$ grid, the tournament and the input chain. First, we show that colouring a tournament of size $< k$ forces, in a.e. G, a $\Theta(n)$ shortage of blank edges (between L and T); this cannot be compensated by colouring edges of other types which are $o(n)$ altogether. Since T is the only k-node tournament, the C-colouring must use it. Now, each C'-link with blank edges can occur at most b_i times in a.e.

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

random graph. Requiring a total of $\sum b_i$ blank edges forces the colouring to use each type exactly b_i times. This strategy crucially depends on G being random.

Let the graph G produced by the reduction R be C-coloured. Let $\widetilde{T}$, $L_{\widetilde{T}}$, and $U_{\widetilde{T}}$ be respectively its sets of blank nodes, looped (non-blank) and unlooped nodes connected to every node in $\widetilde{T}$.

Any link with a blank non-loop edge falls into one of the following disjoint *directed* types: $\mathbf{v},\mathbf{j},\mathbf{i},\kappa,\mathbf{c}$.

The following constraints (i)–(iv) on these types are enforced by the spots in C'.

(i) Each 3-node induced subgraph of $\widetilde{T}$ (and thus the whole $\widetilde{T}$) is a tournament; let $\widetilde{t}_1$ be its sink node.

(ii) Non-blank nodes have at most one link of each of the types: $\kappa,\mathbf{i},\mathbf{j},\mathbf{k},\mathbf{l}$, incoming and outgoing $\mathbf{v},\mathbf{h}$.

(iii) All $\mathbf{i},\mathbf{j},\mathbf{k},\mathbf{l}$ run from $U_{\widetilde{T}}$ to $L_{\widetilde{T}}\cup\{\widetilde{t}_1\}$, $\mathbf{v},\mathbf{h}$ within $L_{\widetilde{T}}\cup\{\widetilde{t}_1\}$, κ single B or Y–B from $U_{\widetilde{T}}$ to $\widetilde{T}$.

(iv) $\mathbf{c}$-links are L–T doubles with a blank edge to T (in two places it is from T).

Put $b_0=k$, $b_1=b_2=b_3=b_4=2p^2$, $b_5=$ the number of double edges L to T, and $b=\sum b_i$. Let $\widetilde{b}_0$ be the number of blanks on $\widetilde{T}$ and $\widetilde{b}_1,\widetilde{b}_2,\dots,\widetilde{b}_5$ be the number of edges of types $\mathbf{v},\mathbf{i},\mathbf{j},\kappa$, and $\mathbf{c}$, respectively. Let $\widetilde{b}=\sum_i\widetilde{b}_i$ be the number of all blank edges.

Proposition 6.1. *Any colouring, respecting* (i)–(iv) *and* $\widetilde{b}\geqslant b$, *of a.e.* $G\in\mathcal{N}$ *has* $\widetilde{b}_i=b_i$ *and* $\widetilde{T}=T$. □

Proof. By (i)–(iv), no non-T node has >2 outgoing blank non-$\mathbf{c}$ edges.
Let $|\widetilde{T}|=k'=k-\Delta$. Then, in a.e. G,

(1) $\Delta\geqslant 0$, since by Corollary 3.3 $k'\leqslant k$,

(2) $<(3/4)^{k'}2n$ nodes are connected to all nodes of any given k'-node set B (by item 2(b) of Remark 3.1).

Any colouring of $G\in\mathcal{N}$ has $b_5-\widetilde{b}_5>\Delta n/9$. To have $\widetilde{b}\geqslant b$, one has to compensate with extra blank edges of other types, whose total number is $<4(3/4)^{k'}n$. This is possible only if

$$4\left(\frac{3}{4}\right)^{k'}n>\frac{\Delta n}{9},$$

that is,

$$\left(\frac{4}{3}\right)^{\Delta}>\left(\frac{4}{3}\right)^{k}\cdot\frac{\Delta}{36},$$

whence $\Delta>k$ for large enough k, which is a contradiction. It follows that $|\widetilde{T}|=k$, and by uniqueness of the k-node tournament, $\widetilde{T}=T$. Thus $U_{\widetilde{T}}=U_T$ and $L_{\widetilde{T}}=L_T$. The number of blank edges of each type $\mathbf{v},\mathbf{i},\mathbf{j},\kappa$, by (iii) is at most $b_1=b_2=b_3=b_4=2p^2$. Finally, the number of loops on T is k. Thus $\widetilde{b}_i\leqslant b_i$, as claimed. □

6.2. Forcing of a monotone order on the input chain and of the toroidal grid

We exhibit the decreasing codes order of v_i as follows. For each i, we choose $l(i)$ and colour a κ-link $(v_i,t_{l(i)})$. The existence of b_4 κ-edges implies that every $v_i\in U_T$ has a κ-link. The conditions described next imply that the colours of links from v_i,z_{i+1},v_{i+1} to T indicate that the code of v_{i+1}

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

with respect to t_j is at most that of v_i for $j > l(i)$ (highest bits), and strictly less for $j = l(i)$. Thus, the input nodes form an acyclic **I**-linked path in decreasing order of their codes. Note that $x > y$ as integers represented as ternary k-digit strings, if and only if for some $l \leqslant k$ and all $j > l$ digits $x_j \geqslant y_j$ and $x_l > y_l$. Recall that a node in $U_T \cup L_T$ is connected to every node in T. The code of v_i with respect to t_j is copied onto $[z_{i+1}, t_j]$ and will be compared in $[z_{i+1}, t_j, v_{i+1}]$.

Colouring the code links. All edges t_j to v_i and *down* edges (single edge to t_j) from v_i are red if not down and $j > l(i)$, blank if $j = l(i)$, or yellow otherwise. The code of v_i with respect to t_j is **11**, **01** or **10** to reflect v_i having a down, up or double edge to t_j. For $j > l(i)$, this ternary code is exhibited on the z_{i+1}–t_j-link by colouring its edges yellow, red or green respectively. For $j = l(i)$, **10** is encoded on the z_{i+1}–t_j-link if v_i has a code **11** with respect to t_j, otherwise **01**. For $j < l(i)$, the yellow z_{i+1}–t_j-link encodes an **11**. Now we can, in $[z_{i+1}, v_{i+1}, t_j]$, uniformly for all i, j, prohibit the codes of v_{i+1} from exceeding those encoded at z_{i+1}.

Toroidal grid. First, we prove the group Γ generated by permutations $\mathbf{v}, \mathbf{h}'$ is commutative. Indeed, consider a **vi**-path x–o (a **v**-link from x followed by an **i**-link to o), a $\mathbf{jv}^{-1}$-path o–y, and an **ij**-path x–z. By our C', they will require an **l**-link x–o, a **k**-link o–y, and an **h**-link x–z; the **lk**-path requires an **h**-link x–y and $\mathbf{i}, \mathbf{l}$ carry the same direction bit **s**. Depending on **s**, the **h**-links at x are both *outgoing* or both *incoming*. By C', x cannot have two such links, thus $y = z$, $\mathbf{h}' = \mathbf{vijv}^{-1}$, so $\mathbf{h}'\mathbf{v} = \mathbf{vij} = \mathbf{vh}'$. □

Claim. $\mathbf{v}, \mathbf{h}'$-links induce a connected toroidal grid on V.

Proof. The **I**-chain spans the whole grid $V := L_T \cup U_T \cup \{z_1\}$. Thus $z_1\Gamma = L_T$: any $u \in L_T$ is accessible from z_1 as $u = \mathbf{h}'^i\mathbf{v}^j(z_1)$. Now, $\mathbf{h}'^r(z_1) = \mathbf{v}^s(z_1) = z_1$ for some minimal r, s, since $\mathbf{v}, \mathbf{h}'$ permute the finite V. If $\mathbf{h}'^i\mathbf{v}^j(z_1) = z_1$ then $\mathbf{h}'^i(z_1) = \mathbf{v}^j(z_1) = z_1$ and $r|i$, $s|j$, since blank z_1 is the only node with both $\mathbf{w}, \mathbf{f}$ flags up in **j**. Then, the $\mathbf{h}'^i\mathbf{v}^j(z_1) \leftrightarrow (i, j)$ bijection sets up a Cartesian grid on V and $rs = 2p^2$. By our C', $r, s > 2$, so the grid is either $2p \times p$ or $p \times 2p$. In our (toroidal) RRTP, exactly one **i**-link changes direction at each row of any tiling. Hence s is even and $s = 2r = 2p$. □

6.3. Forcing the correct representation of tiling

Link bits. The grid links (see Figure 2) reflect RRTP fields: trit τ, bit **b**, side **s**, floor **f** and wall **w**. The self-loop colour at a node in L_T carries its τ; now we list the bits carried by its incident links ($+/-$ refers to their one-transition-later/earlier values): $\mathbf{v} : \mathbf{b}+, \mathbf{s}$ (*i.e.* **v**-links carry $\mathbf{b}+, \mathbf{s}$); $\mathbf{i} : \mathbf{s}, \mathbf{f}$; $\mathbf{j} : \mathbf{f}, \mathbf{w}$; $\mathbf{l} : \mathbf{s}, \mathbf{s}+$; $\mathbf{k} : \mathbf{b}$ if idle, else $\mathbf{w}$; $\mathbf{h} : \mathbf{s}^-, \mathbf{b}$ of both active ends, or $\mathbf{s}^-$ of an idle end. The same triple $(\tau, \mathbf{b}, \mathbf{s})$ can encode a state (if $\mathbf{s} \neq \mathbf{s}^-$) or a symbol (if $\mathbf{s} = \mathbf{s}^-$).

Link colouring. Blanks and loops distinguish all link types (except **k** from **l** that together have 12 edge patterns); $\mathbf{v}, \mathbf{i}, \mathbf{j}$ each have a blank edge with (R/G/Y/A) reverse. **h**-links have no blank edges. Each row of the torus has one active and one wall **h**-link connected by two chains of idle links. Idle **h** have a red edge directed toward the active **h**, and a green reverse. Active links carry $(\mathbf{s}^-, \mathbf{b}_1, \mathbf{b}_2)$ and have a red (if and only if $\mathbf{s}^- = \Leftarrow$) or else a green. If $\mathbf{b}_1 \neq \mathbf{b}_2$, **h** has a yellow edge pointing to $\mathbf{b} = 1$. If $\mathbf{b}_1 = \mathbf{b}_2$, **h** is monochromatic, double if and only if $\mathbf{b} = 1$, else rightward single. If red single with red loops, it is the wall **h**-link.

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

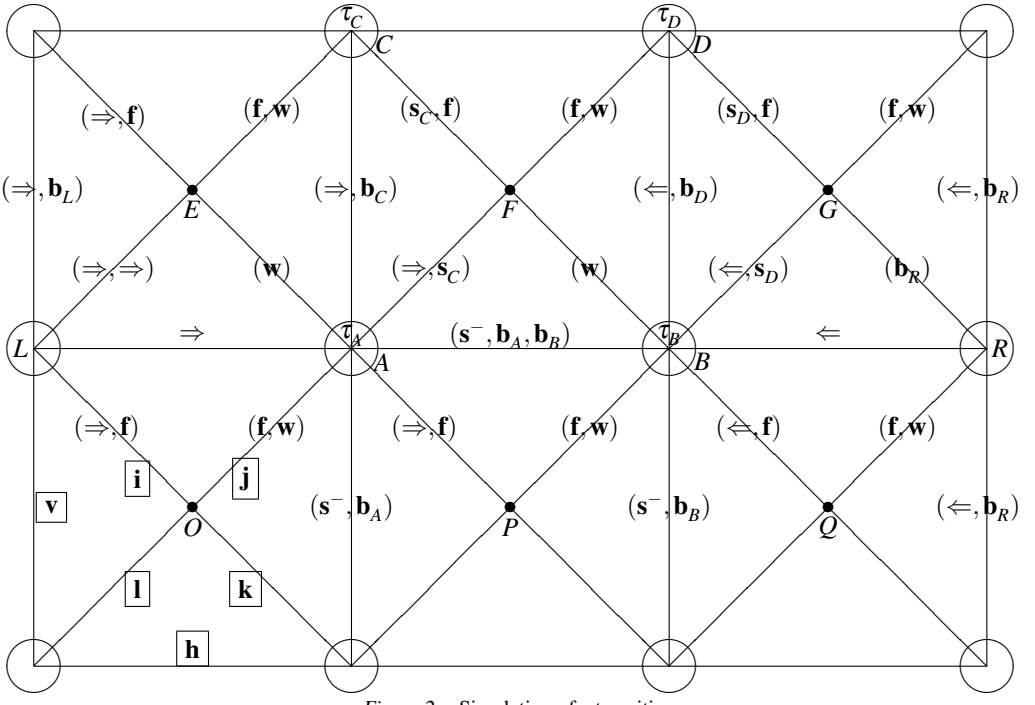

Figure 2. Simulation of a transition.

Link chains. The **i**–**j** and **j**–**k** chains propagate the floor and the wall flags respectively from the origin z_1 to its entire row and column. The wall nodes (in z_1's column) represent the symbol D of the RRTP, and their left neighbours the symbol '0'. They are oriented 'back-to-back' and connected by a special wall **h**-link. It requires the wall-flag up at $\mathbf{j},\mathbf{k}$-links and *vice versa*. Adjacent idle **h**-links match directions. Thus, chains of leftward pointing and rightward pointing idle **h**-links can only meet at a *unique* active **h**-link.

RRTP compliance: the base case. We now tell how *C*-spots force copying the RRTP input onto the grid's floor row (the reduction algorithm *R*, Section 5, step (iv)). On the floor's **j**–**i**-triangles (*e.g.* PBQ in Figure 2), the presence of edges between unlooped ends (PQ) is copied onto its G/R/Y trit (*e.g.* τ_B). Floor's **i**-link at active B has a Y-loop (used for the $*$-symbol and for the starting *e*-state). This precludes edges between *P* and *Q*. This is the leftmost pair of disconnected successive input nodes on the **I**-chain: others to the left are blocked by rightward **h**-links (via rightward **i**-links). The left (rightward directed) segment has trits symbols from $\{0',1',*'\}$, it ends at the active **h**-link, whose right node encodes the starting state. The right segment acts similarly except that only the rightward edge (P,Q) is used and mapped into R/G trit colours.

Induction on rows. In the figure, nodes *A* and *B* are active, their incident edges carry the RRTP triples $(\tau_A,\mathbf{s}_A,\mathbf{b}_A),(\tau_B,\mathbf{s}_B,\mathbf{b}_B)$. Since *A* and *B* have opposite **s** values in the current step, the incoming $\mathbf{s}^-$ are the same so that $\mathbf{s}^-_A=\mathbf{s}^-_B=:\mathbf{s}^-$ as shown in the figure. Now, the **h**-link carries $(\mathbf{s}^-,\mathbf{b}_A,\mathbf{b}_B)$. The triangle *ABC* with the **h**-link on *AB* determines $\mathbf{b}_C$ on the **v**-link *AC* and τ_C at *C*. The triangle *ABF* imposes $\mathbf{s}_C$ on the **l**-link *AF*. Node *D* transition is similar. This ensures the representations of that tile's top symbols at the next row. The **i**-links *FC* and *GD* copy $\mathbf{s}_C$ and $\mathbf{s}_D$

https://doi.org/10.1017/S0963548318000123 Published online by Cambridge University Press

respectively. R is idle so its **v**-link's $\mathbf{b}_R$ is copied from the incoming **v**-link (via the **k**-link). This structure of computation and data flow is similar in each square.

Proof of Proposition 5.5. Recall that a.e. graph G sampled by the reduction $R(X, \alpha)$ is in $\mathcal{N}$, *i.e.* has a unique k-node tournament T, $|U_T| = |L_T| + 1 = 2p^2$, and distinct U_T codes. The tiling instance X determines the edges between successive input nodes. We have described the graph G' and its C-colouration that must encode the tiling pattern. The degree of $G' \setminus T$ is $O(1)$. Thus, by the Embedding Lemma 3.6, G' can be di-embedded in a.e. G, *i.e.* $\bar{\lambda}(\exists g : G' \hookrightarrow G) \sim 1$. □

Acknowledgements

We thank Peter Gács for discussions and Marvin Minsky for comments on small universal Turing machines.

References

[1] Ajtai, M. (1996) Generating hard instances of lattice problems. In *STOC '96: 28th Annual ACM Symposium on Theory of Computing*, ACM, pp. 99–108.
[2] Aharonov, D. and Regev, O. (2005) Lattice problems in NP ∩ coNP. *J. Assoc. Comput. Mach.* **52** 749–765.
[3] Angluin, D. and Valiant, L. G. (1979) Fast probabilistic algorithms for Hamilton circuits and matchings. *J. Comput. Syst. Sci.* **18** 155–193.
[4] Babai, L., Erdos, P. and Selkow, M. (1980) Random graph isomorphism. *SIAM J. Comput.* **9** 628–635.
[5] Begelfor, E., Miller, S. D. and Venkatesan, R. (2015) Non-Abelian analogs of lattice rounding. *Groups Complexity Cryptology* **7** 117–133.
[6] Ben-David, S., Chor, B., Goldreich, O. and Luby, M. (1989) On the theory of average case complexity. In *STOC '89: 21st Annual ACM Symposium on Theory of Computing*, ACM, pp. 204–216.
[7] Blass, A. and Gurevich, Y. (1995) Matrix transformation is complete for the average case. *SIAM J. Comput.* **24** 3–29.
[8] Blum, M. and Micali, S. (1984) How to generate cryptographically strong sequences of pseudo random bits. *SIAM J. Comput.* **13** 850–864.
[9] Bollobás, B. (2001) *Random Graphs*, second edition, Cambridge University Press.
[10] Bollobás, B. (2004) *Extremal Graph Theory*, Dover.
[11] Goldreich, O., Goldwasser, S. and Micali, S. (1986) How to construct random functions. *J. Assoc. Comput. Mach.* **33** 792–807.
[12] Gurevich, Y. (1987) Complete and incomplete randomized NP problems. In *SFCS '87: 28th Annual Symposium on Foundations of Computer Science*, IEEE, pp. 111–117.
[13] Gurevich, Y. and Shelah, S. (1987) Expected computation time for Hamilton path problem. *SIAM J. Comput.* **16** 486–502.
[14] Gurevich, Y. (1990) Matrix decomposition is complete for the average case. In *IEEE FOCS*. **2**, 802–811.
[15] Gurevich, Y. (1991) Average case complexity. *J. Comput. System Sci.* **42** 346–398.
[16] Hajnal, A. and Szemerédi, E. (1970) Proof of a conjecture of P. Erdős. In *Combinatorial Theory and its Applications, II*, North-Holland, pp. 601–623.
[17] Ikeno, S. (1958) A 6-symbol 10-state universal Turing machine. In *Proc. Inst. of Electrical Communications*, Tokyo. (As cited and described in [29].)
[18] Impagliazzo, R. and Levin, L. A. (1990) No better ways to generate hard NP instances than picking uniformly at random. In *FOCS '90: 31st Annual Symposium on Foundations of Computer Science*, IEEE, pp. 812–821.

[19] Jao, D., Miller, S. D. and Venkatesan, R. (2009) Expander graphs based on GRH with an application to elliptic curve cryptography. *J. Number Theory* **129** 1491–1504.
[20] Karp, R. (1976) The probabilistic analysis of some combinatorial search algorithms. In *Algorithms and Complexity* (J. F. Traub, ed.), Academic Press, pp. 1–19.
[21] Karp, R., Lenstra, J. K., McDiarmid, C. J. H. and Rinnoy Kan, A. H. G. (1985) Probabilistic analysis. In *Combinatorial Optimization: Annotated Bibliographies* (M. O'hEigeartaigh, J. K. Lenstra and A. H. G. Rinnoy Kan, eds), Wiley.
[22] Kierstead, H. A. and Kostochka, A. V. (2008) A short proof of the Hajnal–Szemerédi theorem on equitable colouring. *Combin. Probab. Comput.* **17** 265–270.
[23] Johnson, D. (1984) The NP-completeness column: An ongoing guide. *J. Alg.* **5** 284–299.
[24] Lagarias, J. C. and Odlyzko, A. M. (1983) Solving low density subset sum problems. In *24th Annual Symposium on Foundations of Computer Science*, IEEE, pp. 1–10.
[25] Lenstra, A. K. and Lenstra, H. W. (1991) *The Development of the Number Field Sieve*, Springer.
[26] Levin, L. A. (1986) Average case complete problems. *SIAM J. Comput.* **15** 285–286.
[27] Levin, L. A. (2003) The tale of one-way functions. *Prob. Inform. Transm.* **39** 92–103.
[28] Micciancio, D. and Regev, O. (2004) Worst case to average case reductions using Gaussian measures. In *FOCS '04: Annual IEEE Symposium on Foundations of Computer Science*, IEEE, pp. 372–381.
[29] Minsky, M. L. (1967) *Computation: Finite and Infinite Machines*, Prentice Hall.
[30] Shamir, A. (1982) A polynomial algorithm for breaking the basic Merkle–Hellman cryptosystem. In *FOCS '82: 23rd Annual Symposium on Foundations of Computer Science*, IEEE, pp. 145–152.
[31] Venkatesan, R. and Levin, L. A. (1988) Random instances of a graph coloring problem are hard. In *STOC '88: 20th Annual ACM Symposium on Theory of Computing*, ACM, pp. 217–222.
[32] Venkatesan, R. and Rajagopalan, S. (1992) Average case intractability of matrix and diophantine problems. In *STOC '92: 42nd Annual ACM symposium on Theory of Computing*, ACM, pp. 632–642.
[33] Yao, A. C. (1982) Theory and applications of trapdoor functions. In *FOCS '82: 23rd Annual Symposium on Foundations of Computer Science*, IEEE, pp. 80–91.
[34] Wang, J. (1995) Average case completeness of a word problem for groups. In *STOC '95: 27th Annual ACM symposium on Theory of Computing*, ACM, pp. 325–334.