A Measurement-based Algorithm for Graph Colouring

Michael Epping and Tobias Stollenwerk

German Aerospace Center (DLR), Linder Höhe, 51147 Cologne, Germany

We present a novel algorithmic approach to find a proper vertex colouring of a graph with d colours, if it exists. We associate a d-dimensional quantum system with each vertex and the initial state is a mixture of all possible colourings, from which we obtain a random proper colouring of the graph by measurements. The non-deterministic nature of the quantum measurement is tackled by a reset operation, which can revert the effect of unwanted projections. As in the classical case, we find that the runtime scales exponentially with the number of vertices. However, we provide numerical evidence that the average runtime for random graphs scales polynomially in the number of edges.

A proper vertex colouring, often simply called a colouring of a graph, is an assignment of colours to vertices of a graph g, such that no adjacent vertices have the same colour [1]. A colouring that uses d different colours is called a d-colouring and a graph for which a d-colouring exists is called d-colourable. The chromatic number $\chi(g)$ is the smallest number d such that g is d-colourable. See Figure 1 for a graph with chromatic number 3.

It is hard to compute $\chi(g)$. In fact finding $\chi(g)$ is one of Karp's 21 famous NP-complete problems [2, 3, 4, 5], which means that every problem in NP can be solved by calling an oracle version of the graph colouring a polynomial number of times. And even when considering only planar graphs with vertex degree at most four and $d = 3$ different colours the problem remains NP-complete [6]. Various problems of practical significance are naturally linked to the graph colouring problem. This includes planning and scheduling problems as they occur in industrial settings like manufacturing [7] and cost optimization [8]. In the present paper we introduce an algorithm that finds a graph colouring on a quantum computer via measurements. While measurements play a central role in our algorithm and throughout the analysis in this paper, we also use gates. Thus our approach lies in between the gate model paradigm and the framework of measurement-based quantum computation [9].

Michael Epping: Michael.Epping@dlr.de

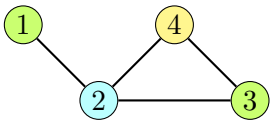


Figure 1: An example of a 3-colourable but not 2-colourable graph.

The paper is structured as follows. We begin by introducing the main idea of the algorithm and present an overview of all relevant steps in Section 1. Afterwards, in Section 2, we cover the technical details of the involved measurements before we present how to handle "unwanted" measurement outcomes in Section 3. We discuss the runtime in Section 4 and give a conclusion in Section 5.

1 The algorithm

The main idea of our algorithm is as follows. Let the number of different colours be d. We associate a d-dimensional qudit with each vertex, such that each colour corresponds to a canonical basis state. Initially we start with a state that contains all colourings, valid and invalid, with nonzero probability. In particular we can start with white noise, also called the completely mixed state. We iteratively colour the edges of the graph by measurements in the canonical basis and a reset operation $\mathcal{B}$ which is necessary due to the stochastic nature of the measurement. The measurement outcome on each qudit is a colour which we assign to the corresponding vertex. This colouring properly colours some of the edges. We call them *established* and keep track of which edges are established already. The reset operation should undo the undesired pro-

jection on the edges which are not established, yet. It must necessarily affect the whole system. The idea is to uniformly shift probability from the measured colouring to all other colourings which respect the established edges. We repeat to ask "Which edges are coloured correctly?" and scrambling the state until we obtain a proper colouring of the graph.

1.1 Basic Concepts and Definitions

We introduce basic concepts before we describe the steps of the algorithm. Let the simple and undirected graph G be given by a set of vertices $V = \{1, 2, ..., n\}$ and a set of edges $E = \{e_1, e_2, ..., e_m\}$. We use d-dimensional qudit states $|\alpha\rangle_v$ to encode the colouring of a vertex $v \in V$ with colour $\alpha \in \{0, \dots, d-1\}$. The corresponding d-dimensional Pauli Operators are defined as

$$\begin{aligned} X_v &:= \sum_{\alpha=0}^{d-1} \omega^n \, |\alpha + 1 \bmod d\rangle_v \, \langle\alpha|_v \\ \text{and } Z_v &:= \sum_{\alpha=0}^{d-1} \omega^n \, |\alpha\rangle_v \, \langle\alpha|_v \end{aligned} \tag{1}$$

with $\omega = \mathrm{e}^{2\pi \mathrm{i}/d}$. In the following, we will drop the vertex index if the meaning should be clear from the context.

We will use the letter c to denote a colouring, i.e. an assignment of colours to all vertices. We usually interpret c as an integer number, where the v-th base-d digit $c_v \in \{0, \dots, d-1\}$ is the colour of the vertex v[1]. The corresponding quantum state is a product state of n qudit states

$$|c\rangle := \bigotimes_{v \in V} |c_v\rangle_v \tag{2}$$

The analysis of the algorithm will be simplified by reinterpreting the overall projection of the measurement in the Z-basis as a sequence of projections onto the subspaces of proper colouring for each edge, followed by the projection onto the measured colouring c. The Hermitian observable $\mathcal{A}^{(e)}$, which answers whether the edge e has a valid colouring, is given by its spectral decomposition

$$\mathcal{A}^{(e)} := P_{=}^{(e)} - P_{\neq}^{(e)}, \tag{3}$$

[1] E.g. $c = (0,1,0,2)\cdot(3^0, 3^1, 3^2, 3^3)^T = 57$ for the colouring with $d = 3$ in Figure 1.

with

$$P_{\circ}^{(\{v,w\})} := \mathbb{1}_{V\setminus\{v,w\}} \otimes \sum_{\substack{\alpha,\beta=0 \\ \alpha\circ\beta}}^{d-1} |\alpha\rangle\langle\alpha|_v \otimes |\beta\rangle\langle\beta|_w \,, \tag{4}$$

where the placeholder $\circ \in \{=, \neq\}$. We denote the set of all edges which are already established with $E_{\neq} \subset E$. We further define the projector

$$P := \prod_{e \in E_{\neq}} P_{\neq}^{(e)}. \tag{5}$$

onto the subspace $\mathcal{H}_P$ where all the established edges in $E_{\neq}$ are coloured properly. Note, that although P explicitly depends on the set of already established edges $E_{\neq}$, we will use this implicit notation P in favour of better readability.

Consider the state with uniform probabilities for all proper colourings on $E_{\neq}$, except for some special colouring c for which the probability is $p_c \in [0, 1]$. We call this state

$$\rho_{\text{ideal}}(P, c, p_c) := p_c \, |c\rangle\langle c| + (1 - p_c) \frac{P - |c\rangle\langle c|}{\operatorname{tr} P - 1}. \tag{6}$$

As we will see later, the goal of the reset operation $\mathcal{B}$ is to spread the probability among all proper colourings on $E_{\neq}$. We introduce the reset operation $\mathcal{B}$ by its action on a colouring state $|c\rangle \in \mathcal{H}_P$

$$\mathcal{B}(|c\rangle\langle c|) = \rho_{\text{ideal}}(P, c, p_c). \tag{7}$$

A state with uniform probabilities, like the initial white noise, is produced for $p_c = \frac{1}{\operatorname{tr} P}$. We interpret p_c as the probability that $\mathcal{B}$ acts like the identity. One may decrease p_c by measuring if the output state is $|c\rangle$ or not and repeating the process in the first case. However, here we accept that our algorithm will not make progress with probability p_c and the runtime increases accordingly. A possible implementation of $\mathcal{B}$ with $p_c = \frac{1}{2}$ is discussed in Section 3.

1.2 Overview of the Algorithm Steps

We now have the necessary definitions to describe the steps involved in our algorithm. We discuss more of the technical details afterwards. A pseudo code summary is given in Algorithm 1.

Algorithm 1 Measurement-based vertex colouring.

procedure VERTEXCOLOURING($g = (V, E)$, d)
▷ Find a proper vertex colouring for the graph g.
$n \leftarrow |V|$
$\rho \leftarrow \mathbb{1}/d^n$
$E_{\neq} \leftarrow \emptyset$
for $k = 1, 2, ..., K$ **do** ▷ Try K times
$c \leftarrow$ measure Z on each vertex
for $(u, v) \in E \backslash E_{\neq}$ **do** ▷ Update $E_{\neq}$
if $c_u \neq c_v$ **then**
$E_{\neq} \leftarrow E_{\neq} \cup \{(u, v)\}$
$k \leftarrow 1$
end if
end for
if $E_{\neq} = E$ **then** ▷ All edges established
return c
else
if $k = K$ **then** ▷ Probably not d-colourable
return no colouring
end if
$\rho \leftarrow \mathcal{B}(|c\rangle\langle c|)$ ▷ Perform reset operation
end if
end for
end procedure

1. Initialize all vertices with white noise, i.e. the density matrix $\mathbb{1}/d$.

2. Measure all vertices in the Z-basis. Let c denote the measurement outcome.

3. Add all edges which c colours properly to the set $E_{\neq}$.

4. If $E_{\neq} = E$ output the colouring c. Else apply $\mathcal{B}$, see below for more details. Retry to find a proper colouring from step 2 again.

5. If K trials fail, we conclude that no valid colouring exists. The error probability for this conclusion is bounded by

$$p_{\text{fail}} \leq (1 - p_{\min})^{p_c K}, \tag{8}$$

where the bound $p_{\min} \leq p_{\neq}^{(e)}$ will be discussed below.

2 The measurement

We now discuss technical details of the algorithm. We start by looking into the success probability of the measurement. While the reset operation is quite generic, this is the graph-colouring-specific part of our algorithm.

Although we described the algorithm with a simple Z-measurement above, it is convenient to imagine preceding $\mathcal{A}^{(e)}$-measurements on all edges e. We call the measurement *successful* for the edge e if the outcome is that the two vertices connected by the edge e have different colours. For now we assume that $\mathcal{A}^{(e)}$ is performed on an ideal state $\rho_{\text{ideal}}(P, c, p_c)$, see Eq. (6), with $p_c = \frac{1}{\operatorname{tr} P}$. Note that the initial state $\mathbb{1}/d^n$ is of this form (for $E_{\neq} = \{\}$). We will discuss the effect of $p_c \neq \frac{1}{\operatorname{tr} P}$ below.

Let $\chi(g, d)$ denote the chromatic polynomial of g evaluated at d colours, which is the number of proper d-colourings of g. Furthermore let $g[S] = (V, S)$ be the graph where the set of edges E is replaced with S. Note that

$$\operatorname{tr} P = \chi(g[E_{\neq}], d). \tag{9}$$

The success probability of $\mathcal{A}^{(e)}$ on any pair of vertices $e = \{v, w\}$, $v, w \in V$ is the ratio of proper colourings for the established graph with and without the edge e added, i.e.

$$p_{\neq}^{(e)} = \frac{\chi(g[E_{\neq} \cup \{e\}], d)}{\chi(g[E_{\neq}], d)}, \tag{10}$$

which can be re-expressed as

$$p_{\neq}^{(e)} = 1 - \frac{\chi(g[E_{\neq}]/e, d)}{\chi(g[E_{\neq}], d)} \tag{11}$$

using the fundamental reduction theorem. Here $/e$ denotes the contraction of e into a single vertex. If the graph is d-colourable, then there are at least d proper colourings. We use this trivial lower bound

$$p_{\neq}^{(e)} \geq p_{\min} := \frac{1}{d^{n-1}} \tag{12}$$

to determine the number of tries K, which affects the runtime of our algorithm. Better bounds might be possible. However, since the focus of this work is the quantum part of the algorithm we leave the exploration of these possibilities to future research.

After a failed $\mathcal{A}^{(f)}$ measurement on edge f, the success probability of $\mathcal{A}^{(e)}$ changes to

$$p_{\neq}^{(e)} = \frac{\chi(g[E_{\neq} \cup \{e\}]/f, d)}{\chi(g[E_{\neq}]/f, d)}. \tag{13}$$

At all points during the algorithm the state ρ is diagonal and lives in $\mathcal{H}_P$. We can think of the effect of the '='-measurement outcome on the "failed" edge f as a disturbance of the probabilities $\rho_{\alpha\alpha}$ on $\mathcal{H}_P$. Before we discuss how to recover $\rho_{\text{ideal}}(P, c, p_c)$, let's consider the example in Figure 2 which illustrates the non-local effects of the failed measurement.

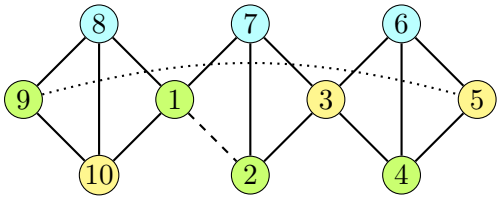


Figure 2: Let $d = 3$. Imagine that the solid line edges have been established already, so we know the neighbouring vertices have different colours. However, we found 1 and 2 to have the same colour. This implies that 5 and 9 have different colours (dotted line), which is not true for a proper colouring of this graph. Therefore the reset operation needs to be global.

Finally, for values of p_c other than $p_c = \frac{1}{\operatorname{tr} P}$ the success probability $p_{\neq}^{(e)}$ for an edge $e = \{v, w\}$ approximately changes to $p_{\neq}^{(e)}(p_c) \approx (1 - p_c)p_{\neq}^{(e)}$, see Appendix A.1.

3 The reset operation $\mathcal{B}$

It is the task of the reset operation $\mathcal{B}$ to redistribute the probability among all $\rho_{\alpha\alpha}$ on $\mathcal{H}_P$, i.e. to decrease p_c in $\rho_{\text{ideal}}(P, c, p_c)$. From a high level view, one could consider $\mathcal{B}$ as a black box whose input is a colouring c which is proper on the edges $E_{\neq}$ and the output is another colouring with this property. All outputs are equally likely, except that with probability p_c we obtain c again.

Although one could imagine different ways to implement $\mathcal{B}$, we decided on an approach based on the quantum Zeno effect [10], where the time evolution due to a Hamilton operator is periodically interrupted by projective measurements. This confines the evolution of the state to the subspace corresponding to the measurement outcome. We exploit this phenomenon to make sure that our mixing operation cannot invalidate edges that have been established already. In the following, we describe the implementation of the reset operation $\mathcal{B}$. For a more elaborate derivation, please find more details in Appendix B.

We define a $d^n \times d^n$ dimensional Hamiltonian H by its matrix elements for the x-th row and the y-th column

$$H_{xy}(\phi) := \begin{cases} \phi & \text{if } x = c \veebar y = c \\ 0 & \text{else,} \end{cases} \tag{14}$$

where $\veebar$ denotes the exclusive or. We denote the short time period in between the N_Z projective measurements

$$\tau := \frac{1}{N_Z}. \tag{15}$$

The probability p_Z to remain in $\mathcal{H}_P$, also called the *survival probability*, scales like

$$p_z = 1 - \mathcal{O}\left(\frac{1}{N_Z^2}\right). \tag{16}$$

The quantum Zeno effect emerges for large N_Z.

Instead of directly implementing $H(\phi)$, the time evolution $\mathrm{e}^{-\mathrm{i}H(\phi)\tau}$ can be implemented via the unitary operation $U^{\epsilon(\phi)}$. The exponent is

$$\epsilon(\phi) := \frac{2\tau\sqrt{d^n - 1}}{\pi}\phi \tag{17}$$

and the base is the unitary

$$U := (\mathbb{1} - 2\,|\tilde{c}\rangle\langle\tilde{c}|)(\mathbb{1} - 2\,|c\rangle\langle c|), \tag{18}$$

which consists of reflections along $|c\rangle$ and

$$|\tilde{c}\rangle := \frac{1}{\sqrt{2}}\,|c\rangle - \frac{\mathrm{i}}{\sqrt{2}}\frac{1}{\sqrt{d^n - 1}}\sum_{\alpha \neq c}|\alpha\rangle\,. \tag{19}$$

Note the similarity with the Grover iteration [11]. Because the eigenvalues of U are i, $-$i and 1, we can exactly implement $U^{\epsilon(\phi)}$ via quantum phase estimation using two ancillary qubits [12], see Figure 3.

The Zeno dynamics are given by the effective Hamiltonian [13, 14]

$$H_Z(\phi) := PH(\phi)P \tag{20}$$

and the effective time evolution

$$U_Z(\phi) := P\mathrm{e}^{-\mathrm{i}H_Z(\phi)}. \tag{21}$$

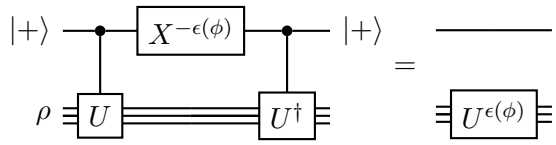


Figure 3: A circuit to implement $U^{\epsilon(\phi)}$, see Eqs. (17) and (18), via quantum phase estimation [12]. The first register, the ancilla, has dimension four. All other registers are d-dimensional qudits. A decomposition of U is discussed in Appendix B.

Note that $U_Z(\phi)$ acts unitarily on $\mathcal{H}_P$. We can dephase the state to simply obtain

$$\begin{aligned}\rho_{\text{out}}(\phi) = \quad & \sin^2\left(\phi\sqrt{\operatorname{tr} P - 1}\right)\frac{P - |c\rangle\langle c|}{\operatorname{tr} P - 1} \\ & + \cos^2\left(\phi\sqrt{\operatorname{tr} P - 1}\right)|c\rangle\langle c|\,.\end{aligned} \tag{22}$$

Now if we had a good estimate for $\operatorname{tr} P$ we could choose ϕ accordingly. But because we cannot make this assumption in our case of the graph colouring algorithm, we will choose ϕ randomly from $[0, \zeta]$, for some $\zeta \in \mathbb{R}$ with $\zeta\sqrt{\operatorname{tr} P} \gg 1$. This results in

$$\rho_{\text{out}} \approx \frac{1}{2}\frac{P - |c\rangle\langle c|}{\operatorname{tr} P - 1} + \frac{1}{2}|c\rangle\langle c|\,, \tag{23}$$

i.e. $\rho_{\text{ideal}}(P, c, p_c)$ with $p_c = \frac{1}{2}$. Algorithm 2 gives a pseudo code summary of the presented reset operation.

Algorithm 2 Reset operation with Zeno-effect.

procedure $\mathcal{B}(|c\rangle\langle c|, N_Z, \zeta, g, E_{\neq}, d)$
▷ Increase probability of other states in $\mathcal{H}_P$.
$n \leftarrow |V|$
$\rho \leftarrow |c\rangle\langle c|$
$\phi \leftarrow \text{random}(0, \zeta)$
$\epsilon \leftarrow \frac{2\sqrt{d^n - 1}}{N_Z \pi}\phi$
for $k = 1, 2, ..., N_Z$ **do**
$\rho \leftarrow U^{\epsilon}\rho U^{-\epsilon}$ ▷ Evolve state, see Figure 3
for $e \in E_{\neq}$ **do** ▷ Measure $\mathcal{A}^{(e)}$ $\forall e \in E_{\neq}$
$\rho \leftarrow P_{\neq}^{(e)}\rho P_{\neq}^{(e)} + (\mathbb{1} - P_{\neq}^{(e)})\rho(\mathbb{1} - P_{\neq}^{(e)})$
end for
end for
return ρ
end procedure

As already mentioned, one can imagine many alternative approaches to implement the incoherent quantum operation $\mathcal{B}$. Consider for example the POVM given by elements

$$\begin{aligned} F_\alpha :=& P \bigotimes_{v \in V} |+_{\alpha_v}\rangle\langle+_{\alpha_v}| P \\ \text{and } \bar{F} :=& \mathbb{1} - P, \end{aligned} \tag{24}$$

where we used

$$|+_\alpha\rangle := Z^\alpha \frac{1}{\sqrt{d}} \sum_{\beta=0}^{d-1} |\beta\rangle \tag{25}$$

with the post-measurement state

$$\tilde{\mathcal{B}}(\rho) = \sqrt{\bar{F}}\rho\sqrt{\bar{F}} + \sum_{\alpha=0}^{d^n-1} \sqrt{F_\alpha}\rho\sqrt{F_\alpha}. \tag{26}$$

This is an X-measurement restricted s.t. established edges are not invalidated. It completely erases the unwanted information gained from a "failed" measurement in the Z-basis. Of course, for any alternative approach one has to evaluate the runtime and practicality. The interested reader may find a failed attempt in this regard in Appendix C. The given examples should merely illustrate that different implementations are possible within the basic approach of our algorithm.

4 Runtime and numerical studies

In order to estimate the quality of our algorithm, we present numerical investigations of the algorithm runtime in this section. These investigations should merely serve as basis for more elaborate numerical analysis of the algorithm which we leave for future work. First, we consider an example which proves particularly challenging for our algorithm. Second, we examine a set of random graphs which are rather hard to color and in order to assess bounds on the algorithm performance. As a last step we discuss the relation of different contributions to the algorithm runtime.

4.1 A particularly hard graph to color

The worst case runtime of our algorithm depends on the lower bound on the success probability $p_{\neq}^{(e)}$ for all $e \in E$, see Eq. (12). This bound leads to a runtime $\mathcal{O}(d^n)$ and hypothetical improvements on the bound would reduce the worst-case runtime accordingly. The success probability can indeed go to zero exponentially in the number of vertices. In order to investigate this we construct a graph which contains an edge which is

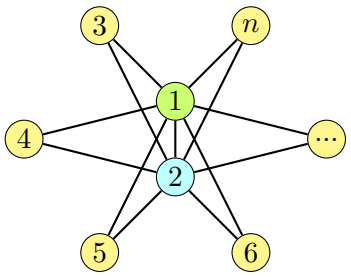


Figure 4: A family of graphs for which the success probability on the $\{1,2\}$-edge is low, because adding this edge removes many possibilities to colour the graph. The edges are $E = \{\{1,2\}\} \cup \{\{1,v\},\{2,v\} \mid 3 \leq v \leq n\}\}$. The chromatic number is $\chi(g) = 3$ for all $n \geq 3$.

particularly hard to colour for our algorithm, see Figure 4. All graphs of this form have chromatic number $\chi(g) = 3$. Let $d = 3$ for this example. Imagine that all edges except $\{1,2\}$ are established. If any two of the outer vertices have different colours, then vertices 1 and 2 have the same colour. Now adding the $\{1,2\}$-edge we loose all these colourings. This leads to the low success probability

$$p_{\neq}^{(\{1,2\})} = \frac{1}{1+2^{n-3}} = \frac{1}{1+2^{\lfloor \frac{m-3}{2} \rfloor}}, \tag{27}$$

see Appendix A.3 for the derivation.

4.2 Random Graphs

In addition to the special problem instance above, we investigate the average success probability for random graphs, generated via random vertex degree distributions [15], i.e. we choose the vertex degree of each vertex uniformly random from $\{1,2,...,d+1\}$. In order to use rather hard instances, we restrict the chromatic number to $\chi = d$, since the complexity (of backtrack-type algorithms) peaks at $d = \chi$ or $d = \chi + 1$ [16] (See Appendix D for details). The result of this investigation is depicted in Figure 5a and Figure 5b, which show the success probability of our algorithm for random graphs of different sizes.

In addition to the success probability for a single edge we can also look at the average time required to establish all edges in g. We again do this via random graphs. It is important to note that graph distributions may contain many easy instances. This can lead to wrong conclusions about the usefulness of the algorithm. We therefore try to avoid easy instances of the graph colouring problem by requiring $\chi(g) \geq \log_2 n$ and by only using the optimal number of colours $d = \chi(g)$. See Appendix D for more details on our numeric studies and our efforts to improve the quality of the assessment. We label the edges e_1, e_2, ..., e_m. For simplicity, we assume that they are established one edge at a time and in the given order. In practice this would imply that we would only add the next edge in the list to the set of established edges and ignore all other edges. Of course our algorithm can establish many edges simultaneously, which can only decrease the runtime. Under this assumption the success probability of the i-th edge e_i is

$$p_i := p_{\neq}^{(e_i)} \text{ with } E_{\neq} = \{e_j \mid j < i\}. \tag{28}$$

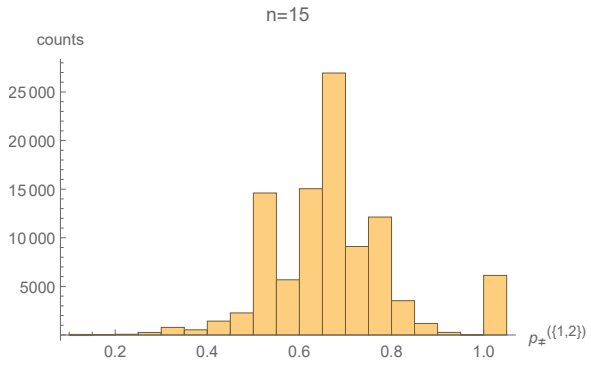
Neglecting the costs of the reset operation, we can approximate the runtime of our algorithm by summing the expected runtime for each individual edge. We denote this sum with

$$\Sigma := \sum_{k=1}^{m} \frac{1}{p_k}. \tag{29}$$

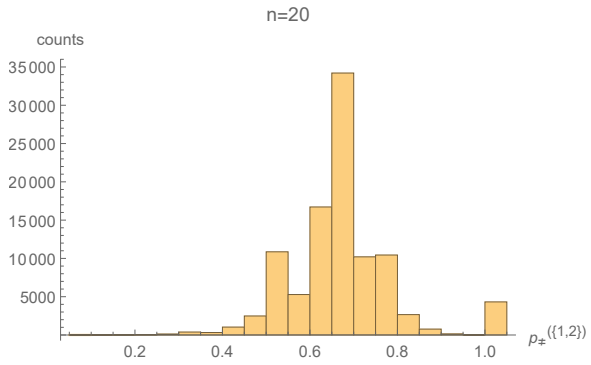
Note that in general Σ depends on the ordering of the edges. In some sense our algorithm is similar to randomly guessing a proper colouring. However, the crucial difference between our algorithm and simple guessing is that for us, because of the reset operation, a wrong guess is not fatal. The runtime of the naive guessing approach, which is also the expected runtime of trying all possible colourings systematically, is

$$\Pi := \prod_{k=1}^{m} \frac{1}{p_k} = \frac{d^n}{\chi(g,d)}. \tag{30}$$

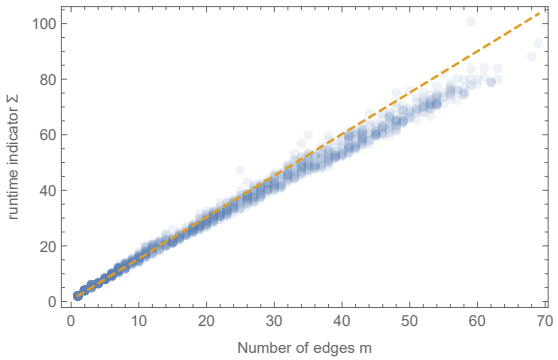
Of course there are much more sophisticated approaches, but for a first analysis we stick to Π as a reference. Since it is unlikely that efficient classical algorithms can approximate $\chi(g)$, we do not compare the runtime of our algorithm to heuristics [17]. The ratio of the two runtimes is shown in Figure 5d for samples of random graphs. Figure 5c shows the roughly linear dependence of Σ on the number of edges m for the same random graphs. We estimate the runtime without the fixed order of the edges in Appendix A.2. Some special cases, for example bipartite graphs, are known to be easy to colour. This is also true for our algorithm: With $d = 2$ the success probability of an edge can be either 1, if both vertices are connected already, 0 if no colouring exists, or $\frac{1}{2}$



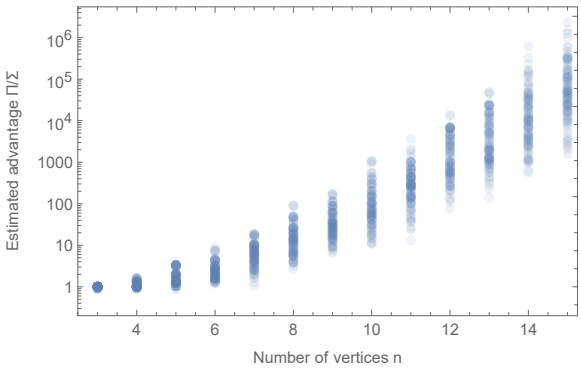
(a) $n = 15$ vertices



(b) $n = 20$ vertices



(c) The approximated runtime Σ as a function of the edge count. The line corresponds to $\frac{3}{2}m$



(d) An estimated advantage compared to naive guessing

Figure 5: Numerical evidence for a linear average runtime of our algorithm. (a), (b): The probability of a successful $\mathcal{A}^{(\{1,2\})}$-measurement, see Eq. (10), for $d = 3$ on 1×10^5 random graphs with $n = 15$ and $n = 20$ vertices, chromatic number 3 and maximal vertex degree 4. The random graphs are generated via random vertex degree distributions. (c), (d): The approximated runtime Σ as a function of the edge count m and the ratio of the "runtime" of naive guessing Π (see Eq. (30)) and the runtime indicator Σ (see Eq. (29)). Each plot point corresponds to a randomly generated graph g and $d = \chi(g) = 3$, see Appendix D.

else. Also the case $\Delta = d = 3$, seems to have a constant lower bound on the success probability $p_{\min} = \frac{2}{7}$. We verified this for all graphs with less than ten vertices.

4.3 Runtime Discussion

We allowed the reset operation $\mathcal{B}$ to act as identity with probability p_c. This increases the runtime of the overall algorithm by a factor of roughly $\frac{1}{1-p_c}$, which is 2 for the implementation of $\mathcal{B}$ which we presented. The size of the circuit for the reset operation is polynomial in the number of vertices. We can perform $\frac{1}{1-p_z}$ reset operations $\mathcal{B}$ before we expect errors due to the deviation of p_z from 1. Note that the reset operation starts from the classical state $|c\rangle\langle c|$, which can be restored so that these errors are not fatal. In conclusion, the runtime of the algorithm is dominated by the $\mathcal{A}$-measurement, not the reset operation $\mathcal{B}$.

Since both classical and quantum algorithms for the graph colouring problem scale exponentially in the number of vertices [18, 19, 20], there is little hope for gaining a substantial speed-up. However, we provide numerical evidence that our approach works well for some instances of the problem for which other approaches are not well suited.

If there is quantum speed-up it is interesting to understand its origin. Apart from the specific implementation of the incoherent operation $\mathcal{B}$, entanglement plays no role in our algorithm. Also we make heavy use of mixed states and even start from white noise. So what is "quantum" about our approach? Put differently, which obstacles would one run into when trying to implement our algorithm on a classical computer? The reset operation together with the Z-measurement samples, given a specific proper colouring c as the input, from the uniform distribution of all proper colourings of the graph formed by the established edges. With a classical computer this task does not look simpler than the original problem of finding any proper colouring. It is also not an option to keep the diagonal entries of ρ in memory, because these are too many already for moderate sizes of n. Thus we identify the reset operation as the crucial quantum part of our algorithm, even though it is an incoherent operation. And we believe that it is impossible to efficiently implement the reset operation without exploiting coherence internally in some form. It is worth noting in this context that once a single proper colouring of the graph is obtained, the reset operation $\mathcal{B}$ can be used to sample from all proper colourings in $\mathcal{O}(1)$. This is a problem with it's own applications [21]. The basic idea of non-deterministic measurements and resets is applicable beyond the graph colouring problem which we discussed in the present paper. Applications to other problems of practical relevance will be presented elsewhere.

5 Conclusion

We presented a novel approach to solve graph colouring problems with quantum computers. In the course of this, we employ both concepts from measurement-based as well as gate-based quantum computing. The two main steps in the algorithm are the measurement operation which mainly drives the runtime, and a reset operation which corrects for undesired measurement outcomes. To support the main contribution of this work, which is the presentation of the novel approach, we provided a preliminary runtime analysis both using analytical as well as numerical means. Future work could extend the algorithm to other problems beyond graph colouring and elaborate on the analysis of the performance as well as improvements thereof.

Acknowledgements

M.E. and T.S. acknowledge funding from the DLR project EQUATE. M.E. thanks Davide Orsucci, Elisabeth Lobe and Peter Ken Schuhmacher for helpful discussions.

References

[1] Sven Oliver Krumke and Hartmut Noltemeier. *Graphentheoretische Konzepte und Algorithmen*. 1615-5432. Vieweg+Teubner Verlag, Wiesbaden, 3 edition, 2012. ISBN 978-3-8348-1849-2. DOI: https://doi.org/10.1007/978-3-8348-2264-2. German.

[2] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceed-*

ings of the Third Annual ACM Symposium on Theory of Computing, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery. ISBN 9781450374644. DOI: 10.1145/800157.805047. URL https://doi.org/10.1145/800157.805047.

[3] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2˙9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

[4] L. A. Levin. Universal sequential search problems. *Probl. Peredachi Inf.*, 9:115–116, 1973. URL http://mi.mathnet.ru/ppi914.

[5] B.A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984. DOI: 10.1109/MAHC.1984.10036.

[6] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. ISSN 0304-3975. DOI: https://doi.org/10.1016/0304-3975(76)90059-1. URL https://www.sciencedirect.com/science/article/pii/0304397576900591.

[7] Michael Streif, Sheir Yarkoni, Andrea Skolik, Florian Neukart, and Martin Leib. Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm. *Phys. Rev. A*, 104:012403, Jul 2021. DOI: 10.1103/PhysRevA.104.012403. URL https://link.aps.org/doi/10.1103/PhysRevA.104.012403.

[8] Tobias Stollenwerk, Stuart Hadfield, and Zhihui Wang. Toward quantum gate-model heuristics for real-world planning problems. *IEEE Transactions on Quantum Engineering*, 1:1–16, 2020. DOI: 10.1109/TQE.2020.3030609.

[9] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. Measurement-based quantum computation on cluster states. *Phys. Rev. A*, 68:022312, Aug 2003. DOI: 10.1103/PhysRevA.68.022312. URL https://link.aps.org/doi/10.1103/PhysRevA.68.022312.

[10] B. Misra and E. C. G. Sudarshan. The Zeno's paradox in quantum theory. *Journal of Mathematical Physics*, 18(4):756–763, April 1977. DOI: 10.1063/1.523304.

[11] Lov K. Grover. A fast quantum mechanical algorithm for database search. *28th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 212–219, May 1996.

[12] L Sheridan, D Maslov, and M Mosca. Approximating fractional time quantum evolution. *Journal of Physics A: Mathematical and Theoretical*, 42(18):185302, apr 2009. DOI: 10.1088/1751-8113/42/18/185302. URL https://doi.org/10.1088/1751-8113/42/18/185302.

[13] P. Facchi and S. Pascazio. Quantum zeno subspaces. *Phys. Rev. Lett.*, 89:080401, Aug 2002. DOI: 10.1103/PhysRevLett.89.080401. URL https://link.aps.org/doi/10.1103/PhysRevLett.89.080401.

[14] P Facchi and S Pascazio. Quantum zeno dynamics: mathematical and physical aspects. *Journal of Physics A: Mathematical and Theoretical*, 41(49):493001, oct 2008. DOI: 10.1088/1751-8113/41/49/493001. URL https://doi.org/10.1088/1751-8113/41/49/493001.

[15] Mark Newman. *Networks: An Introduction*. Oxford University Press, Oxford, 2010.

[16] Zoltan Mann. Complexity of coloring random graphs: An experimental study of the hardest region. *Journal of Experimental Algorithmics*, 23:1–19, 03 2018. DOI: 10.1145/3183350.

[17] M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, January 1976. ISSN 0004-5411. DOI: 10.1145/321921.321926. URL https://doi.org/10.1145/321921.321926.

[18] Marek Cygan, Holger Dell, Daniel Lokshtanov, D'niel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as cnf-sat. In *2012 IEEE 27th Conference on Computational Complexity*, pages 74–84, 2012. DOI: 10.1109/CCC.2012.36.

[19] Andris Ambainis, Kaspars Balodis, Janis Iraids, Martins Kokainis, Krisjanis Prusis, and Jevgenijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1783–1793. SIAM, 2019. DOI: 10.1137/1.9781611975482.107. URL https://doi.org/10.1137/1.9781611975482.107.

[20] Kazuya Shimizu and Ryuhei Mori. Exponential-time quantum algorithms for graph coloring problems. In Yoshiharu Kohayakawa and Flávio Keidi Miyazawa, editors, *LATIN 2020: Theoretical Informatics*, pages 387–398, Cham, 2020. Springer International Publishing. ISBN 978-3-030-61792-9.

[21] Alan Frieze and Eric Vigoda. A survey on the use of markov chains to randomly sample colourings. *Combinatorica*, 01 2007. DOI: 10.1093/acprof:oso/9780198571278.003.0004.

[22] B. Bollobás. The chromatic number of random graphs. *Combinatorica*, 8:49–55, march 1988. DOI: 10.1007/BF02122551. URL https://doi.org/10.1007/BF02122551.

A Technical details for the $\mathcal{A}^{(e)}$-measurements

In this appendix we discuss the $\mathcal{A}^{(e)}$-measurements in more detail. A lot of this can be described in the language of stochastics and graph theory, and quantum information theory only occasionally.

A.1 The effect of $p_c \neq \frac{1}{\operatorname{tr} P}$

For the sake of a simple presentation we assumed that all states have uniform or zero probability in the mixed state. Our implementation of the reset operation $\mathcal{B}$ accomplishes this only with the exception of the initial colouring c. In this section we describe how this fact affects the success probability of establishing an edge. The success probability of $e = \{v, w\}$ in Eq. (10) holds for $p_c = \frac{1}{\operatorname{tr} P}$ for which the ideal state in Eq. (6) is

$$\rho_{\text{ideal}}(P, c, \frac{1}{\operatorname{tr} P}) = \frac{1}{\operatorname{tr} P} |c\rangle\langle c| + \left(1 - \frac{1}{\operatorname{tr} P}\right) \frac{P - |c\rangle\langle c|}{\operatorname{tr} P - 1} = \frac{P}{\operatorname{tr} P}. \tag{31}$$

In this state the derivation of the success probability is simple, because the probabilities of all colourings in $\mathcal{H}_P$ are identical. Now rewrite

$$\rho_{\text{ideal}}(P, c, p_c) = \left(\frac{p_c \operatorname{tr} P - 1}{\operatorname{tr} P - 1}\right) |c\rangle\langle c| + (1 - p_c) \frac{\operatorname{tr} P}{\operatorname{tr} P - 1} \rho_{\text{ideal}}(P, c, \frac{1}{\operatorname{tr} P}) \tag{32}$$

and calculate the success probability

$$\begin{aligned}
p_{\neq}^{(e)}(p_c) &= \operatorname{tr}\left(\rho_{\text{ideal}}(P, c, p_c) P\right) && (33)\\
&\quad \downarrow \text{ Insert Eq. (32), and express } \operatorname{tr} P = \chi(g[E_{\neq}]).\\
&= \frac{(1 - p_c)\chi(g[E_{\neq}])}{\chi(g[E_{\neq}]) - 1} p_{\neq}^{(e)} + \left(\frac{p_c \chi(g[E_{\neq}]) - 1}{\chi(g[E_{\neq}]) - 1}\right)(1 - \delta_{c_v c_w}) && (34)\\
&\quad \downarrow \text{ For large } \chi(g[E_{\neq}])\\
&\approx (1 - p_c) p_{\neq}(e) + p_c(1 - \delta_{c_v c_w}). && (35)
\end{aligned}$$

We are mostly interested in the success probability of edges $e = \{v, w\} \in E \backslash E_{\neq}$ that are not established, yet. Then $\delta_{c_v c_w} = 1$ and the success probability roughly gets scaled by a factor $(1 - p_c)$.

A.2 The runtime

In the main text we considered the expression Σ in Eq. (29) as an upper bound on the runtime. There we assumed that the edges are established in a given order. However, in our algorithm the edges can be established in any order. This improves the runtime but it also makes the calculation more complicated. In this section we give a recursive formula for this more accurate calculation.
Starting from some set of established edges $E_{\neq}$, the expected number of measurement attempts $\sigma(g, E_{\neq})$ until the graph g is coloured can be expressed recursively. Again we assume $p_c = \frac{1}{\operatorname{tr} P}$ for now. This implies that all colourings c which respect $E_{\neq}$ are equally likely. Then the runtime is

the runtime given the established edges of c plus one:

$$\sigma(g, E_{\neq}) = 1 + \frac{1}{\operatorname{tr} P} \sum_{\substack{c \\ |c\rangle \in \mathcal{H}_P}} \sigma(g, \{\{v, w\} \,|\, c_v \neq c_w\}) \tag{36}$$

$q(E_{\neq}, F)$ is the probability to establish only the edges in the set F.

$$= 1 + \sum_{F \subset E \setminus E_{\neq}} q(E_{\neq}, F)\sigma(g, E_{\neq} \cup F) \tag{37}$$

The $(F = \emptyset)$-term equals the lefthand side, so we take it out of the sum.

$$= \frac{1}{1 - q(E_{\neq}, \emptyset)} \left(1 + \sum_{\substack{F \subset E \setminus E_{\neq} \\ F \neq \emptyset}} q(E_{\neq}, F)\sigma(g, E_{\neq} \cup F) \right) \tag{38}$$

The recursion ends with $\sigma(g, E) = 0$. Because we start the algorithm with no established edges, the overall runtime is

$$\sigma(g) := \sigma(g, \emptyset). \tag{39}$$

A.3 Example

We calculate the success probability of the $\{1, 2\}$-edge in the graph shown in Figure 4 when all other edges are established already. For the whole graph g, including the edge $\{1, 2\}$, the chromatic polynomial can be derived as follows. The first node can be coloured with one of d colours. The second node can be coloured with $(d - 1)$ colours. Then there are $(d - 2)$ colours left for each of the outer nodes and we obtain

$$\chi(g, d) = d(d - 1)(d - 2)^{n-2}. \tag{40}$$

If we remove the edge $\{1, 2\}$, we can do the same analysis but now we distinguish two cases. The number of colours available to the outer nodes depends on whether 1 and 2 have the same colour or not. Summing the two possibilities gives

$$\chi(g[E\backslash\{e\}], d) = \underbrace{d(d - 1)^{n-2}}_{\text{same colour}} + \underbrace{d(d - 1)(d - 2)^{n-2}}_{\text{different colour}}. \tag{41}$$

The success probability is the ratio of the two:

$$p_{\neq}(\{1, 2\}) = \frac{\chi(g, d)}{\chi(g[E\backslash\{e\}], d)} = \frac{1}{(d - 1)^{n-3}(d - 2)^{2-n} + 1} \tag{42}$$

for $d = 3$

$$= \frac{1}{2^{n-3} + 1}. \tag{43}$$

Eq. (43) is the worst-case probability for this example.

B Technical details for $\mathcal{B}$ based on the Zeno-effect

Let c be a proper colouring on $E_{\neq}$. The initial state of the system for the reset operation is

$$\rho_{\text{in}} = |c\rangle\langle c| \,. \tag{44}$$

We also define

$$|c_{\perp}\rangle := \frac{1}{\sqrt{d^n - 1}} \sum_{\substack{\alpha=0 \\ \alpha \neq c}}^{d^n - 1} |\alpha\rangle \,, \tag{45}$$

where the naming stems from $\langle c|c_\perp\rangle = 0$. Let the Hamilton operator $H(\phi)$ be a $d^n \times d^n$-matrix with the entry

$$H_{xy}(\phi) := \begin{cases} \phi & \text{if } x = c \veebar y = c \\ 0 & \text{else,} \end{cases} \tag{46}$$

in the x-th row and y-th column, where $\veebar$ denotes the exclusive or. It will be convenient later on to express

$$H(\phi) = \phi\sqrt{d^n - 1}\left(|c\rangle\langle c_\perp| + |c_\perp\rangle\langle c|\right). \tag{47}$$

Suppose we apply the Hamiltonian $H(\phi)$ for a short time period

$$\tau = \frac{1}{N_Z}, \tag{48}$$

followed by a projective measurement of P vs. $\mathbb{1} - P$, and repeat this N_Z times. Let $\mathcal{H}_P$ denote the subspace on which P acts like the identity. For large N_Z this leads to a quantum Zeno effect. It might not be obvious how to apply $H(\phi)$ directly. The same time evolution can be achieved via a unitary gate U, because

$$\begin{aligned}
U(\phi)^\tau &= \left(\mathrm{e}^{-\mathrm{i}H(\phi)}\right)^\tau && (49)\\
&\quad \text{Pull } \tau \text{ into exponential function, insert a factor of 1.} \\
&= \exp\left(-\mathrm{i}H\left(\frac{\pi}{2\sqrt{d^n-1}}\right)\frac{2\sqrt{d^n-1}}{\pi}\phi\tau\right) && (50)\\
&\quad \text{Introduce } \epsilon(\phi) := \tfrac{2\tau\sqrt{d^n-1}}{\pi}\phi \\
&= \exp\left(-\mathrm{i}H\left(\frac{\pi}{2\sqrt{d^n-1}}\right)\epsilon(\phi)\right) && (51)\\
&\quad \text{Define } U := \exp\left(-\mathrm{i}H\left(\tfrac{\pi}{2\sqrt{d^n-1}}\right)\right) \\
&= U^{\epsilon(\phi)}. && (52)
\end{aligned}$$

We can express U as a composition of two reflections,

$$\begin{aligned}
U &= \exp\left(-\mathrm{i}H\left(\frac{\pi}{2\sqrt{d^n-1}}\right)\right) && (53)\\
&\quad \text{Rewrite } H(\phi) \text{ using } |c\rangle \text{ and } |c_\perp\rangle, \text{ see Eq. (47).} \\
&= \exp\left(-\mathrm{i}\frac{\pi}{2}(|c\rangle\langle c_\perp| + |c_\perp\rangle\langle c|)\right) && (54)\\
&\quad (|c\rangle\langle c_\perp| + |c_\perp\rangle\langle c|)^{2k} = |c\rangle\langle c| + |c_\perp\rangle\langle c_\perp| \\
&\quad (|c\rangle\langle c_\perp| + |c_\perp\rangle\langle c|)^{2k+1} = |c\rangle\langle c_\perp| + |c_\perp\rangle\langle c| \\
&= \mathbb{1} - I(|c\rangle\langle c_\perp| + |c_\perp\rangle\langle c|) - (|c\rangle\langle c| + |c_\perp\rangle\langle c_\perp|) && (55)\\
&\quad |\tilde{c}\rangle := \tfrac{1}{\sqrt{2}}(|c\rangle - \mathrm{i}\,|c_\perp\rangle) \\
&= (\mathbb{1} - 2\,|\tilde{c}\rangle\langle\tilde{c}|)(\mathbb{1} - 2\,|c\rangle\langle c|). && (56)
\end{aligned}$$

So U is a reflection along $|c\rangle$ followed by a reflection along $|\tilde{c}\rangle$. Let V_c and $V_{\tilde{c}}$ be circuits that prepare the states $|c\rangle$ and $|\tilde{c}\rangle$, respectively, i.e. they fulfill

$$V_c\,|0\rangle = |c\rangle \tag{57}$$

$$\text{and } V_{\tilde{c}}\,|0\rangle = |\tilde{c}\rangle\,. \tag{58}$$

One possibility to implement $V_{\tilde{c}}$ is to use an ancillary qubit (with dimension two). It is convenient to describe the circuit for $c = 0$ and swap the role of $|0\rangle$ and $|c\rangle$ afterwards. First apply a single qubit gate R_a to the ancillary qubit in order to prepare a suitable superposition of $|0\rangle$ and $|1\rangle$. Then apply

a controlled-Fourier gate from the ancillary qubit to the system. A second single qubit gate R_b follows and finishes the desired transformation up to phases which are corrected via

$$\tilde{Z} = |0\rangle\langle 0|_{\text{anc.}} \otimes (|0\rangle\langle 0| - (\mathbb{1} - |0\rangle\langle 0|)i) + |1\rangle\langle 1|_{\text{anc.}} \otimes (|0\rangle\langle 0| + (\mathbb{1} - |0\rangle\langle 0|)i). \tag{59}$$

The single qubit gates are $R_a = \vec{a} \cdot \vec{\sigma}$ and $R_b = \vec{b} \cdot \vec{\sigma}$ with

$$\vec{a} = \frac{1}{\sqrt{2(d^n-1)}} \left(\sqrt{d^n}, 0, \sqrt{d^n-2}\right)^T \tag{60}$$

$$\text{and } \vec{b} = \frac{1}{\sqrt{2(d^n-1)}} \left(\sqrt{d^n - 1 + \sqrt{d^n-1}}, 0, \sqrt{d^n - 1 - \sqrt{d^n-1}}\right)^T, \tag{61}$$

$$\tag{62}$$

respectively. With this choice we have

$$V_0^\dagger |0\rangle \otimes |0\rangle = \tilde{Z} R_b (|0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes F) R_a |0\rangle \otimes |0\rangle \tag{63}$$

Apply R_a.

$$= \tilde{Z} R_b (|0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes F)(a_z |0\rangle + a_x |1\rangle) \otimes |0\rangle \tag{64}$$

Apply controlled-Fourier-gate.

$$= \tilde{Z} R_b (a_z |0\rangle \otimes |0\rangle + a_x |1\rangle \otimes |+\rangle) \tag{65}$$

Apply R_b.

$$= \tilde{Z}(a_z(b_z |0\rangle + b_x |1\rangle) \otimes |0\rangle + a_x(b_x |0\rangle - b_z |1\rangle) \otimes |+\rangle) \tag{66}$$

Collect terms for the $|0\rangle$ and $|1\rangle$ term on the ancilla.

$$= \tilde{Z}(|0\rangle \otimes (a_z b_z |0\rangle + a_x b_x |+\rangle) + |1\rangle \otimes (a_z b_x |0\rangle - a_x b_z |+\rangle)) \tag{67}$$

Factor out the norm.

$$= \tilde{Z}(b_x |0\rangle \otimes (\frac{a_z b_z}{b_x} |0\rangle + a_x |+\rangle) + b_z |1\rangle \otimes (\frac{a_z b_x}{b_z} |0\rangle - a_x |+\rangle)) \tag{68}$$

Insert some vector components of $\vec{a}$ and $\vec{b}$.

$$= \tilde{Z} \frac{1}{\sqrt{2}} \left(b_x |0\rangle \otimes \left(\left(1 - \frac{1}{\sqrt{d^n-1}}\right) |0\rangle + \frac{\sqrt{d^n}}{\sqrt{d^n-1}} |+\rangle \right) \right.$$
$$\left. + b_z |1\rangle \otimes \left(\left(1 + \frac{1}{\sqrt{d^n-1}}\right) |0\rangle - \frac{\sqrt{d^n}}{\sqrt{d^n-1}} |+\rangle \right) \right) \tag{69}$$

Simplify the $|0\rangle$-component.

$$= \tilde{Z} \frac{1}{\sqrt{2}} (b_x |0\rangle \otimes (|0\rangle + |0_\perp\rangle) + b_z |1\rangle \otimes (|0\rangle - |0_\perp\rangle)) \tag{70}$$

Apply $\tilde{Z}$.

$$= b_x |0\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - i |0_\perp\rangle) + b_z |1\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - \mathrm{i} |0_\perp\rangle) \tag{71}$$

Now we have a product state.

$$= (b_x |0\rangle + b_z |1\rangle) \otimes |\tilde{0}\rangle . \tag{72}$$

Then we can compose a circuit for U via

$$\begin{aligned} U =& (\mathbb{1} - 2 |\tilde{c}\rangle\langle\tilde{c}|)(\mathbb{1} - 2 |c\rangle\langle c|) \\ =& V_{\tilde{c}} Z' V_{\tilde{c}}^\dagger V_c Z' V_c^\dagger, \end{aligned} \tag{73}$$

with

$$Z' := \mathbb{1} - 2 |0\rangle\langle 0| . \tag{74}$$

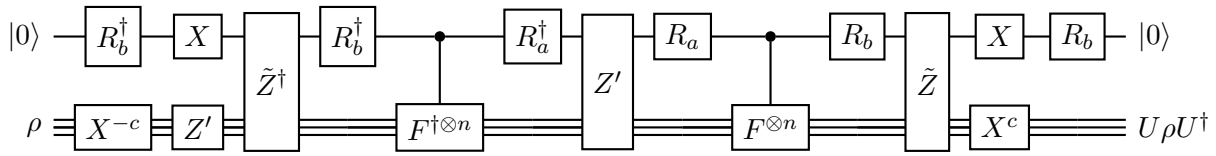


Figure 6: A circuit diagram of a possible decomposition of U, see Eq. (73). The first register is a qubit (two-dimensional), the rest are d-dimensional qudits. The diagonal gates $\tilde{Z}$ and Z' are defined in Eqs. (59) and (74), respectively. A circuit for controlled-U can be obtained by replacing each gate with a controlled version of it.

The resulting circuit is sketched in Figure 6. Because

$$U^4 = \mathbb{1}, \tag{75}$$

the eigenvalues of U are in $\{1, \mathrm{i}, -1, -\mathrm{i}\}$ (they are i, $-$i and $(d^n - 2)$-times 1). Therefore we need only two ancillary qubits to implement $U^{\epsilon(\phi)}$ via quantum phase estimation, see Figure 3. With the described circuit for the time evolution associated with the Hamilton operator $H(\phi)$ we can implement the Zeno effect. The Zeno dynamics are given by

$$H_Z(\phi) := PH(\phi)P \tag{76}$$

$$\text{and } U_Z(\phi) := P\mathrm{e}^{-\mathrm{i}H_Z(\phi)}. \tag{77}$$

Note that P cuts out the part of $H(\phi)$ which acts on $\mathcal{H}_P$. This part has exactly the same structure as the original Hamiltonian of the whole system. Now the dimension of $\mathcal{H}_P$, $\operatorname{tr} P$, plays the role of the dimension d^n. There is no dependence on d or n left in H_Z and U_Z. The effect of $U_Z(\phi)$ on $|c\rangle$ is

$$U_Z(\phi)\,|c\rangle = P \exp\left(-\mathrm{i}PH(\phi)P\right)|c\rangle \tag{78}$$

Express $H(\phi)$ via $|c\rangle$ and $|c_\perp\rangle$, see Eq. (47).

$$= P \exp\left(-\mathrm{i}\phi P\sqrt{d^n-1}(|c\rangle\langle c_\perp| + |c_\perp\rangle\langle c|)P\right)|c\rangle \tag{79}$$

Apply P to these states and normalize by inserting a factor of 1.

$$= P \exp\left(-\mathrm{i}\phi\sqrt{\operatorname{tr} P - 1}(|c\rangle \frac{\sqrt{d^n-1}}{\sqrt{\operatorname{tr} P - 1}}\langle c_\perp| P + \frac{\sqrt{d^n-1}}{\sqrt{\operatorname{tr} P - 1}} P|c_\perp\rangle\langle c|)\right)|c\rangle \tag{80}$$

Evaluate the exponential function for the orthonormal vectors in the argument

$$= P\Big(\cos(\phi\sqrt{\operatorname{tr} P - 1})(|c\rangle\langle c| + |c_\perp\rangle\langle c_\perp|) - \mathrm{i}\sin(\phi\sqrt{\operatorname{tr} P - 1})(|c\rangle \frac{\sqrt{d^n-1}}{\sqrt{\operatorname{tr} P - 1}}\langle c_\perp| P + \frac{\sqrt{d^n-1}}{\sqrt{\operatorname{tr} P - 1}} P|c_\perp\rangle\langle c|)\Big)|c\rangle \tag{81}$$

Apply everything to $|c\rangle$ on the right, using $\langle c|c_\perp\rangle = 0$.

$$= \cos\left(\phi\sqrt{\operatorname{tr} P - 1}\right)|c\rangle - \mathrm{i}\sin\left(\phi\sqrt{\operatorname{tr} P - 1}\right)\frac{\sqrt{d^n-1}}{\sqrt{\operatorname{tr} P - 1}} P|c_\perp\rangle\,. \tag{82}$$

Since we directly measure the state in the Z-basis, we can drop the off-diagonal elements of the density matrix and obtain

$$\rho_{\text{out}}(\phi) = \sin^2\left(\phi\sqrt{\operatorname{tr} P - 1}\right)\frac{P - |c\rangle\langle c|}{\operatorname{tr} P - 1} + \cos^2\left(\phi\sqrt{\operatorname{tr} P - 1}\right)|c\rangle\langle c|\,. \tag{83}$$

Up until now we did not specify the value of ϕ, yet. Ideally we would like to choose ϕ to be

$$\phi_{\text{all}} := \frac{\tan^{-1}\sqrt{\operatorname{tr} P - 1}}{\sqrt{\operatorname{tr} P - 1}} \tag{84}$$

$$\text{or } \phi_{\text{others}} := \frac{\pi}{2\sqrt{\operatorname{tr} P - 1}}. \tag{85}$$

The names are related to the colourings in the output state, because

$$\rho_{\text{out}}(\phi_{\text{all}}) = \frac{P}{\operatorname{tr} P}, \tag{86}$$

$$\text{whereas } \rho_{\text{out}}(\phi_{\text{others}}) = \frac{P - |c\rangle\langle c|}{\operatorname{tr} P - 1}. \tag{87}$$

But because we might not have a good estimate for $\operatorname{tr} P$, we now look into choosing ϕ at random. Note that the time evolution is periodic, i.e.

$$U_Z\left(\phi + \frac{2\pi}{\sqrt{\operatorname{tr} P - 1}}\right) = U_Z(\phi). \tag{88}$$

For large $\operatorname{tr} P$ the time evolution is quickly oscillating. If we choose ϕ uniformly random from $[0, \zeta]$, then we end up with the state

$$\rho_{\text{out}} := \frac{1}{\zeta}\int_0^\zeta \rho_{\text{out}}(\phi)d\phi \tag{89}$$

$$\int_0^b \sin^2(ax)dx = \tfrac{b}{2} - \tfrac{\sin(2ab)}{4a}$$

$$= \left(\frac{1}{2} - \frac{\sin(2\zeta\sqrt{\operatorname{tr} P - 1})}{4\zeta\sqrt{\operatorname{tr} P - 1}}\right)\frac{P - |c\rangle\langle c|}{\operatorname{tr} P - 1} + \left(\frac{1}{2} + \frac{\sin(2\zeta\sqrt{\operatorname{tr} P - 1})}{4\zeta\sqrt{\operatorname{tr} P - 1}}\right)|c\rangle\langle c| \tag{90}$$

$$\approx \frac{1}{2}\frac{P - |c\rangle\langle c|}{\operatorname{tr} P - 1} + \frac{1}{2}|c\rangle\langle c| \tag{91}$$

The approximation is accurate for $\zeta \gg 1$, because

$$\left|\frac{\sin(2\zeta\sqrt{\operatorname{tr} P - 1})}{4\zeta\sqrt{\operatorname{tr} P - 1}}\right| \leq \frac{1}{4\zeta\sqrt{\operatorname{tr} P - 1}}. \tag{92}$$

So we have a fifty-fifty chance of producing a new proper colouring on $E_{\neq}$. Otherwise we are left with the initial state, which will not progress the algorithm. In either case we will not leave $\mathcal{H}_P$ with probability $1 - \mathcal{O}(\frac{1}{N_z^2})$.

C An edge reset operation based on the SWAP gate

In this appendix we discuss an approach to implement $\mathcal{B}$ whose runtime, unfortunately, turns out to scale exponentially in the problem size. We double the size of the register with n ancillary qudits, labelled $n+1, n+2, ..., 2n$, which are initialized with white noise. The operation

$$U_v := \sum_{\alpha,\beta=0}^{d-1} |\alpha\beta\rangle\langle\beta\alpha|_{v,n+v} \tag{93}$$

swaps the vertex v and the corresponding ancillary qudit. The unitary

$$U := \bigotimes_{v\in V} U_v \tag{94}$$

swaps the whole original and the ancillary system. It could be implemented by applying the Hamiltonian

$$\begin{aligned} H :=& \mathrm{i}\log U \\ =& \frac{\pi}{2}(U - \mathbb{1}) \end{aligned} \tag{95}$$

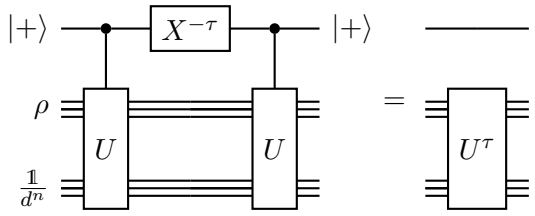
for a time $t = 1$, because the time evolution is

$$U(t) = \mathrm{e}^{-\mathrm{i}tH} \overset{t=1}{=} \mathrm{e}^{\log U} = U. \tag{96}$$

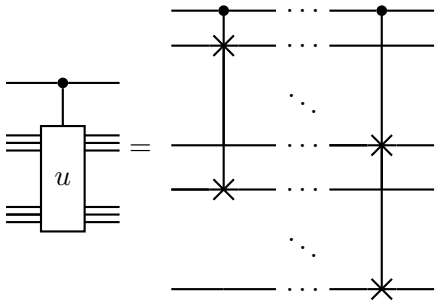
In order to keep established edges, the Hamiltonian can be interrupted with frequent $\mathcal{A}^e$ measurements for all $e \in E_{\neq}$. See [13, 14] for more details on the Zeno dynamics in subspaces. Let the number of measurements be N_Z. They divide the time $t = 1$ in N_Z intervals of length $\tau = \frac{1}{N_Z}$. A circuit to implement the time evolution in these short intervals,

$$U(\tau) = \mathrm{e}^{-\mathrm{i}H\tau} = U^{-\tau}, \tag{97}$$

via quantum phase estimation (QPE) [12] is shown in Figure 7. For large N_Z this procedure effectively



(a) A circuit to implement U^{-r}. The first register is a qubit, all other register are d-dimensional qudits.



(b) The controlled unitary gate in (a) can be decomposed into controlled-SWAP operations.

Figure 7: Circuit diagrams for the reset operation.

implements the Hamiltonian

$$H_Z := PHP \tag{98}$$

and the unitary operation on the $\mathcal{H}_P$-subspace

$$U_Z := P\mathrm{e}^{-\mathrm{i}H_Z}. \tag{99}$$

A state $|c\rangle\langle c| \in \mathcal{H}_P$ of the qudits without the ancillas is mapped to

$$\begin{aligned} &\mathrm{tr}_{n+1\ldots 2n}\, U_Z \left(|c\rangle\langle c| \otimes \frac{\mathbb{1}}{d^n} \right) U_Z^\dagger \\ =&\frac{\mathrm{tr}\, P}{d^n} \rho_{\mathrm{ideal}}(P, c, p_c) + \left(1 - \frac{\mathrm{tr}\, P}{d^n}\right) |c\rangle\langle c| \,. \end{aligned} \tag{100}$$

Repeating the described procedure, starting with fresh ancillary states, the state converges exponentially in the number of repetitions to $\rho_{\mathrm{ideal}}(P, c, p_c)$. However, the scaling with n is unfavourable. We will see this after some calculations.

Note that $U^2 = \mathbb{1}$ implies that the logarithm of the swap operation U is

$$\log U = \frac{\pi i}{2} (\mathbb{1} - U), \tag{101}$$

because

$$\exp\frac{\pi\mathrm{i}}{2}(\mathbb{1}-U) = \sum_{k=0}\frac{\left(\frac{\pi\mathrm{i}}{2}\right)^k(\mathbb{1}-U)^k}{k!} \tag{102}$$

$$(\mathbb{1}-U)^2 = \mathbb{1}-2U+U^2 = 2(\mathbb{1}-U)$$

$$= \mathbb{1} + \sum_{k=1}\frac{\left(\frac{\pi\mathrm{i}}{2}\right)^k 2^{k-1}(\mathbb{1}-U)}{k!} \tag{103}$$

$$\text{Pull } (\mathbb{1}-U) \text{ out of the sum, start sum from } k=0$$

$$= \mathbb{1} + (\mathbb{1}-U)\frac{1}{2}\left(-1+\sum_{k=0}\frac{(\pi\mathrm{i})^k}{k!}\right) \tag{104}$$

$$\mathrm{e}^{\mathrm{i}\pi} = -1$$

$$= \mathbb{1} - (\mathbb{1}-U) = U. \tag{105}$$

In the following we derive an explicit form of U_Z by calculating the exponential function.

$$U_Z = P\exp\left(-\mathrm{i}P\mathrm{i}\log UP\right) \tag{106}$$

$$\text{Eq. (101)}$$

$$= P\exp\left(-\mathrm{i}P\frac{\pi}{2}(\mathbb{1}-U)P\right) \tag{107}$$

$$P \text{ and } PUP \text{ commute.}$$

$$= P\exp\left(-\mathrm{i}\frac{\pi}{2}P\right)\exp\left(-\mathrm{i}\frac{\pi}{2}PUP\right) \tag{108}$$

$$\mathrm{e}^{-\frac{\pi\mathrm{i}}{2}} = -\mathrm{i}.$$

$$= -\mathrm{i}P\exp\left(\frac{\pi\mathrm{i}}{2}PUP\right) \tag{109}$$

$$\text{Use series expansion of the exponential function.}$$

$$= -\mathrm{i}P\sum_{k=0}^{\infty}\frac{1}{k!}\left(\frac{\pi\mathrm{i}}{2}PUP\right)^k \tag{110}$$

$$\text{Split in sums over even and odd numbers.}$$

$$= -\mathrm{i}P\left(\sum_{k=0}^{\infty}\frac{1}{(2k)!}\left(\frac{\pi\mathrm{i}}{2}PUP\right)^{2k} + \sum_{k=0}^{\infty}\frac{1}{(2k+1)!}\left(\frac{\pi\mathrm{i}}{2}PUP\right)^{2k+1}\right) \tag{111}$$

$$(PUP)^4 = (PUP)^2,\ (PUP)^3 = PUP.$$

$$= -\mathrm{i}P + PUP + \mathrm{i}PUPUP \tag{112}$$

We can use this expression to calculate the evolution of the quantum state under the Zeno dynamics,

i.e. Eq. (100) of the main text.

$$\operatorname{tr}_{\text{anc}} U_Z \left|c\right\rangle\!\left\langle c\right| \otimes \frac{\mathbb{1}}{d^n} U_Z^\dagger = \frac{1}{d^n} \operatorname{tr}_{\text{anc}}(-\mathrm{i}P + PUP + \mathrm{i}PUPUP) \left|c\right\rangle\!\left\langle c\right| \otimes \mathbb{1}(\mathrm{i}P + PUP - \mathrm{i}PUPUP) \tag{113}$$

Expand the factors

$$\begin{aligned} = \frac{1}{d^n} \operatorname{tr}_{\text{anc}} (\quad & P \left|c\right\rangle\!\left\langle c\right| P \otimes \mathbb{1} \\ & - \mathrm{i}P \left|c\right\rangle\!\left\langle c\right| \otimes \mathbb{1} PUP \\ & - P \left|c\right\rangle\!\left\langle c\right| \otimes \mathbb{1} PUPUP \\ & + \mathrm{i}PUP \left|c\right\rangle\!\left\langle c\right| P \otimes \mathbb{1} \\ & + PUP \left|c\right\rangle\!\left\langle c\right| \otimes \mathbb{1} PUP \\ & - \mathrm{i}PUP \left|c\right\rangle\!\left\langle c\right| \otimes \mathbb{1} PUPUP \\ & - PUPUP \left|c\right\rangle\!\left\langle c\right| P \otimes \mathbb{1} \\ & + \mathrm{i}PUPUP \left|c\right\rangle\!\left\langle c\right| \otimes \mathbb{1} PUP \\ & + PUPUP \left|c\right\rangle\!\left\langle c\right| \otimes \mathbb{1} PUPUP) \end{aligned} \tag{114}$$

Apply the transformation in each term, followed by tracing out the ancillary system. Note that $\left|c\right\rangle \in \mathcal{H}_P$.

$$\begin{aligned} = \frac{1}{d^n} & (d^n \left|c\right\rangle\!\left\langle c\right| - \mathrm{i} \left|i\right\rangle\!\left\langle i\right| - \operatorname{tr} P \left|c\right\rangle\!\left\langle c\right| + \mathrm{i} \left|c\right\rangle\!\left\langle c\right| + P \\ & - \mathrm{i} \left|c\right\rangle\!\left\langle c\right| - \operatorname{tr} P \left|c\right\rangle\!\left\langle c\right| + \mathrm{i} \left|c\right\rangle\!\left\langle c\right| + \operatorname{tr} P \left|c\right\rangle\!\left\langle c\right|) \end{aligned} \tag{115}$$

Several terms cancel each other out.

$$= \frac{1}{d^n} (d^n \left|c\right\rangle\!\left\langle c\right| + P - \operatorname{tr} P \left|c\right\rangle\!\left\langle c\right|) \tag{116}$$

Factor out $\left|c\right\rangle\!\left\langle c\right|$

$$= \frac{P}{d^n} + \frac{1}{d^n} (d^n - \operatorname{tr} P) \left|c\right\rangle\!\left\langle c\right| \tag{117}$$

identify $\rho_{\text{ideal}}(P, c, p_c)$, see Eq. (6).

$$= \frac{\operatorname{tr} P}{d^n} \rho_{\text{ideal}} \left(P, c, \frac{1}{\operatorname{tr} P}\right) + \left(1 - \frac{\operatorname{tr} P}{d^n}\right) \left|c\right\rangle\!\left\langle c\right| . \tag{118}$$

The evolution of a diagonal mixed state $\rho = \sum_\alpha \rho_{\alpha\alpha} \left|\alpha\right\rangle\!\left\langle\alpha\right|$,

$$\operatorname{tr}_{\text{anc}} U_Z \rho \otimes \frac{\mathbb{1}}{d^n} U_Z^\dagger = \frac{\operatorname{tr} P}{d^n} \rho_{\text{ideal}}(P, c, p_c) + \left(1 - \frac{\operatorname{tr} P}{d^n}\right) \rho, \tag{119}$$

follows by linearity. It remains to show that by applying this evolution repeatedly the state converges to $\rho_{\text{ideal}}(P, c, p_c)$. In the worst case we start the reset operation from a pure state $\left|c\right\rangle\!\left\langle c\right|$. The state will have the form

$$\rho = x \rho_{\text{ideal}}(P, c, \frac{1}{\operatorname{tr} P}) + (1 - x) \left|c\right\rangle\!\left\langle c\right| \tag{120}$$

with the maximal entry

$$\left\langle c\right| \rho \left|c\right\rangle = 1 - \left(1 - \frac{1}{\operatorname{tr} P}\right) x \tag{121}$$

for some $x \in [0, 1]$. The index c is also the position with the maximal deviation from the target state $\rho_{\text{ideal}}(P, c, 1/\operatorname{tr} P)$. The ratio of this deviation for the state before and after the Zeno process,

$$\frac{\left\langle c\right| \rho \left|c\right\rangle - \frac{1}{\operatorname{tr} P}}{\left\langle c\right| \left(\frac{\operatorname{tr} P}{d^n} \rho_{\text{ideal}}(P, c, p_c) + \left(1 - \frac{\operatorname{tr} P}{d^n}\right) \rho\right) \left|c\right\rangle - \frac{1}{\operatorname{tr} P}} = \frac{1 - \left(1 - \frac{1}{\operatorname{tr} P}\right) x - \frac{1}{\operatorname{tr} P}}{\frac{1}{d^n} + \left(1 - \frac{\operatorname{tr} P}{d^n}\right) \left(1 - \left(1 - \frac{1}{\operatorname{tr} P}\right) x\right) - \frac{1}{\operatorname{tr} P}} \tag{122}$$

which can be simplified to

$$= \frac{d^n}{d^n - \operatorname{tr} P}, \tag{123}$$

is a constant. As noted above, this implies that the state converges exponential in the number of repetitions of the Zeno process to the ideal state. Let the number of steps be N_Z in the Zeno process and the number of repetitions of that process be M_Z. The survival probability p_Z, i.e. the probability for the state to remain in the subspace of the established edges, is $p_Z = 1 - \mathcal{O}(N_Z^{-2})$. For a fixed deviation δ from the ideal state, M_Z scales like $\log \delta / \left(\log d^n - \log(d^n - \operatorname{tr} P)\right) \approx \log(\delta) \frac{d^n}{\operatorname{tr} P}$. This is why we consider this a failed attempt to implement $\mathcal{B}$ *efficiently*.

D Details on the numerical studies and the random graph generation

When trying to assess the performance of a graph colouring algorithm one should avoid running it on "easy" instances. We used two methods of generating random graphs in our numerical studies.

Bernoulli random graphs Bernoulli random graphs, also called Erdős-Rényi model, or the usual random graph model $G(n, p)$, are generated if each edge appears with the same probability p. We choose $p = \frac{1}{2}$. In our studies we additionally require $\chi(g) \geq \log_2 n$ and only use the optimal number of colours $d = \chi(g)$. The first restriction on the chromatic number is only relevant for the small instances that we consider, because almost all random graphs have [22]

$$\chi(g) = \left(\frac{1}{2} + o(1)\right) \log\left(\frac{1}{1-p}\right) \frac{n}{\log n}. \tag{124}$$

A python implementation of our random graph generation is shown in Algorithm 3.

Algorithm 3 Bernoulli random graphs.

```
from sage.graphs.graph_coloring import chromatic_number
from sage.graphs.generators.random import RandomGNP
from math import log

def BernoulliRandomGraph(n):
        p = 0.5
        dmin = log(n, 2)
        while (True):
                g = RandomGNP(n, p)
                d = chromatic_number(g)
                if (d >= dmin):
                        return g
```

Vertex-degree distributions We first choose the vertex degree for each vertex uniformly random from $[1, \Delta]$, where $\Delta \in \mathbb{N}$ is the maximal vertex degree. Then we generate a random graph which has these vertex degrees, if possible. We check whether $\chi(g) = d$ for some fixed d and retry the described process until the condition is satisfied. A minimal python implementation of this procedure is given in Algorithm 4.

Algorithm 4 Graphs with random degree sequence.

```
from sage.graphs.graph_coloring import chromatic_number
from sage.combinat.degree_sequences import DegreeSequences
import random

def RandomConfigurationModel(n, max_degree, chi):
        while (True):
                while (True):
                        z = [random.randint(1, max_degree) for x in range(n)]
                        if sum(z) % 2 == 0:
                                break
                g = graphs.DegreeSequenceConfigurationModel(z)
                if (g.has_loops() or g.has_multiple_edges()):
                        continue
                if (chromatic_number(Graph(g.adjacency_matrix(), multiedges=False)
                        continue
                return Graph(g, multiedges = False, loops = False)
```