Design Patterns in Beeping Algorithms (extended abstract)*

A. Casteigts, Y. Métivier, J.M. Robson, and A. Zemmari

LaBRI, University of Bordeaux
`{acasteig, metivier, robson, zemmari}@labri.fr`

Abstract

We consider networks of processes which interact with beeps. In the basic model defined by Cornejo and Kuhn [5], which we refer to as the BL variant, processes can choose in each round either to beep or to listen. Those who beep are unable to detect simultaneous beeps. Those who listen can only distinguish between silence and the presence of at least one beep. Stronger variants exist where the nodes can also detect collision while they are beeping ($B_{cd}L$) or listening (BL_{cd}), or both ($B_{cd}L_{cd}$). Beeping models are weak in essence and even simple tasks are difficult or unfeasible with them.

This paper starts with a discussion on generic building blocks (*design patterns*) which seem to occur frequently in the design of beeping algorithms. They include *multi-slot phases*: the fact of dividing the main loop into a number of specialised slots; *exclusive beeps*: having a single node beep at a time in a neighbourhood (within one or two hops); *adaptive probability*: increasing or decreasing the probability of beeping to produce more exclusive beeps; *internal* (resp. *peripheral*) collision detection: for detecting collision while beeping (resp. listening); and *emulation* of collision detection: for enabling this feature when it is not available as a primitive.

We then provide algorithms for a number of basic problems, including colouring, 2-hop colouring, degree computation, 2-hop MIS, and collision detection (in BL). Using the patterns, we formulate these algorithms in a rather concise and elegant way. Their analyses (in the full version) are more technical, e.g. one of them relies on a Martingale technique with non-independent variables; another improves that of the MIS algorithm in [8] by getting rid of a gigantic constant (the asymptotic order was already optimal).

Finally, we study the relative power of several variants of beeping models. In particular, we explain how *every* Las Vegas algorithm with collision detection can be converted, through emulation, into a Monte Carlo algorithm without, at the cost of a logarithmic slowdown. We prove that this slowdown is optimal up to a constant factor by giving a matching lower bound.

Digital Object Identifier 10.4230/LIPIcs... **Keywords.** Beeping models, Design patterns, Collision detection, Colouring, 2-hop colouring, Degree computation, Emulation.

1 Introduction

Distributed computing is concerned with various assumptions, like the structure of the network (trees, rings, planar graphs, *etc.*) or knowledge available to the nodes (network size, identifiers, port numbering, *etc.*). Another important aspect is the size of messages, which may range from unbounded, to logarithmic size, to constant size.

As a natural goal is to reduce assumptions as much as possible. Typically, when a problem is solved in some strong model, the community strives to solve it in weaker models. In a recent series of works [5, 10, 1, 7, 8, 6], new models were explored that are even weaker than constant size messages. They are called *beeping models*.

* This research has been supported by ANR projects DESCARTES (ANR-16-CE40-0023) and ESTATE (ANR-16-CE25-0009-03). A full version is available on arXiv (`http://arxiv.org/abs/1607.02951`)

In beeping models, the only communication capabilities offered to the nodes are to *beep* or to *listen*. Several variants exist. In [5], a node that beeps is unable to detect whether other nodes have beeped simultaneously. When listening, it can distinguish between silence or the presence of at least one beep, but it cannot distinguish between one and several beeps. In Section 6 of [1], beeping nodes can detect whether other nodes are beeping simultaneously. In [10] and Section 4 of [1], yet another variant is considered where listening nodes can tell the difference between silence, one beep, and several beeps.

In this paper, we denote the ability to detect collision while beeping (internal collision) by B_{cd} and that of detecting collision while listening (peripheral collision) by L_{cd}. The absence of such ability is denoted by B and L, respectively. The existing models can be reformulated using the cartesian product of these capabilities. Hence, the basic model introduced by Cornejo and Kuhn in [5] is BL; the model considered by Afek et al. in [1] (Section 6) and Jeavons et al. in [8] is $B_{cd}L$; and the model considered in [10] and in Section 4 of [1] is BL_{cd}. To the best of our knowledge, $B_{cd}L_{cd}$ was only used in a previous work of the authors [3].

Although some variants are stronger than others, all beeping models remain extremely weak in essence. Yet, they are relevant to account for real-world applications or phenomena. For instance, they reflect the features of a network at the lowest levels (physical and MAC layers), where a node can probe or emit signals, with or without collision detection. At a higher level of abstraction, beeping models also reflect some communication patterns in biology [4, 1, 9].

1.1 Contributions

The contributions of this paper are manifold. As a warm-up, we start by identifying generic building blocks (*design patterns*) which seem to occur often in the design of beeping algorithms. Then we present a number of algorithms for various graph problems which improve upon previous solutions. Finally, we generalise existing emulation techniques for using collision detection if it is not available, and we prove them optimal *w.h.p.* up to a constant factor.

Due to space limitations, this version of the paper omits (in its core) most complexity analyses and some proofs. However, both are available respectively in Appendix A and B of the present paper, as well as in the arXiv version whose reference is given in the first page.

1.1.1 Design patterns.

We identify a number of common building blocks in beeping algorithms, including *multi-slot phases*: the fact of dividing the main loop into a (typically constant) number of slots having specific roles (*e.g.,* contention among neighbours, collision detection, termination detection); *exclusive beeps*: the fact of having a single node beep at a time in a neighbourhood (within one or two hops, depending on the needs); *adaptive probability*: increasing or decreasing the probability of beeping in order to maximise the number of exclusive beeps; *internal* (resp. *peripheral*) collision detection: the fact of detecting collision while beeping (resp. listening); and *emulation* of collision detection: the fact of detecting collisions even when it is not available as a primitive. As we show in the paper, these patterns make it possible to formulate the algorithms in a rather concise and elegant way.

1.1.2 Algorithms and analyses for basic graph problems.

We present, or analyse algorithms for a number of basic graph problems, including colouring, 2-hop colouring, degree computation, Maximal Independent Set (MIS) and 2-hop MIS. Quite

Model	Time (# slots)	Message size	Knowledge	# colours
$B_{cd}L$	$O(\log n + \Delta)$ expected and *w.h.p.*	$\simeq 1$ bit ($B_{cd}L$ beeps)	None	$O(\log n + \Delta)$
$B_{cd}L$	$O\left(K(\log n + \log^2 K)\right)$ *w.h.p.*	$\simeq 1$ bit ($B_{cd}L$ beeps)	Upper bound K on the max degree of G	K

Table 1 Randomised Las Vegas colouring algorithms on graphs with n vertices.

often, the design of algorithms is easier and more natural if collision detection is assumed as a primitive, e.g., in $B_{cd}L_{cd}$ or $B_{cd}L$. Furthermore, emulation techniques such as those described later in this paper enable safe and automatic translations of algorithms into weaker models like BL. For this reason, our algorithms are expressed using whichever model is the most convenient.

First, we present a Las Vegas (i.e. guaranteed result, uncertain time) colouring algorithm in the $B_{cd}L$ model, with time complexity of $O(\log n + \Delta)$ slots *w.h.p.*, where Δ is the maximum degree in G. Its analysis relies on a martingale technique with non-independent random variables, which makes use of a result by Azuma [2] (details in appendix). In fact, the phenomenon is quite ubiquitous in beeping models: the algorithm terminates in the first moment when every node has produced an exclusive beep at least once within its (1-hop) neighbourhood. This stopping time is made more complex by the use of the *adaptive probability* pattern mentioned above. Another algorithm for 2-hop colouring is given, this time in the $B_{cd}L_{cd}$ model, with slot complexity $O(\log n+\Delta^2)$ *w.h.p.* Both algorithms require no knowledge on G. However, both can result in arbitrarily many colours (in fact, one per slot). If the nodes know an upper bound $K \geq \Delta$, a different strategy is proposed that uses at most $K + 1$ colours. However, the slot complexity becomes $O(K(\log n + \log^2 K))$ *w.h.p.* for colouring (trade K for K^2 in the 2-hop variant). Note that this complexity is not thought to be tight. The results are summarised on Table 1.

Based on the observation that degree computation is strongly related to 2-hop colouring, we present an adaptation of the algorithm for this problem, with same slot complexity, that is, $O(\log n + \Delta^2)$ *w.h.p.* In fact, the random process induced by this algorithm is the same as that of colouring, except that it occurs in the *square* of the graph (whence the Δ^2 term). Algorithmically, the main loop contains more specialised slots (e.g., one for peripheral collision reporting), but still a constant number of them, which keeps the asymptotics unchanged. We then turn our attention to the 2-hop MIS problem, which shares common traits and patterns with 2-hop colouring and degree computation and, regarding the high-level purpose of each phase, with the MIS algorithm from [8]. The running time is however shorter than that of 2-hop coulouring and degree computation (and the analysis quite different) due to the fact that exclusive beeps cause whole neighbourhoods to terminate at once. In fact, we prove that the slot complexity of this algorithm is $O(\log n)$ *w.h.p.* with a "reasonable" constant factor of 76. Noteworthily, the number of phases (i.e. iterations of the main loop) for the 2-hop MIS is exactly the same as what the analogue for classical MIS would produce in the square of the graph. As a consequence, our analysis also improves substantially that of the MIS algorithm presented in [8], where a gigantic constant factor (i.e. one larger than e^{25}) is used. An earlier analysis in [11] yielded a better, yet huge constant of 2×10^{11}. Although constant factors are less meaningful in general, the gap in this case is one between practical and unpractical running times. Furthermore, the contribution it not as much in the constant itself than in the analysis techniques that achieve it.

1.1.3 Collision detection and emulation techniques.

Classical considerations on symmetry breaking in anonymous beeping networks, see for example [1] (Lemma 4.1), imply that there is no Las Vegas internal collision detection algorithm in the beeping models BL and BL_{cd}. Likewise, there is no Las Vegas peripheral collision detection algorithm in the beeping models BL and $B_{cd}L$. Since collision detection is required to detect exclusive beeps with certainty, and this pattern is central in most beeping algorithms, this implies that a large range of algorithms cannot exist in a Las Vegas version in these models.

We study the cost of detecting collision when it is not available, typically in BL, and present generic techniques to emulate collision detection probabilistically in order to transform Las Vegas algorithms with collision detection into Monte Carlo algorithms (uncertain result, guaranteed time) in BL. These techniques generalise that of Algorithm 3 in [1], where a similar strategy is encapsulated into the algorithm. We show how, given $0 < \epsilon < 1$, any collision in the neighbourhood of a *given* node can be detected in $O(\log(\frac{1}{\epsilon}))$ slots with error at most ϵ, and similarly it can be detected in $O(\log n)$ slots *w.h.p.* Ensuring that this is true for *any* node requires more time. By union bound, it holds that $O(\log(\frac{n}{\epsilon}))$ slots are sufficient with error ϵ and that $O(\log n)$ slots are sufficient *w.h.p.* We prove that this technique is essentially optimal (asymptotically and up to a constant factor) by giving a matching lower bound. Precisely, we prove that some topologies require $\Omega(\log n)$ slots to break symmetries *w.h.p.* Finally, we provide two generic procedures that can be used in an algorithm to emulate collision detection when it is not available (e.g. in BL). These procedures are `EmulateB`$_{cd}$`inBL()`, to detect collision while beeping, and `EmulateL`$_{cd}$`inBL()`, to detect collision while listening. We illustrate their use in the case of the computation of a MIS given in $B_{cd}L$, thus obtaining a Monte Carlo algorithm in BL.

1.2 Organisation of the paper

In Section 2 we present the model and give further definitions. Section 3 introduces design patterns in a tutorial manner. These patterns are then used in Section 4 to describe the various algorithms. For the sake of readability, the corresponding analyses are put together in Section A. Finally, Section 5 presents our contribution on collision detection and emulation techniques. An extra bibliography is provided in Section **??** on related questions.

2 Network Model and Definitions

We consider a wireless network and we follow definitions given in [1] and [5]. The network is anonymous: unique identifiers are not available to distinguish the processes. Possible communications are encoded by a graph $G = (V, E)$ where the nodes V represent processes and the edges E represent pairs of processes that can hear each other. We denote by Δ the maximum degree of G. The neighbourhood of a vertex v, denoted $N(v)$, is the set of vertices adjacent to v (at distance 1 from v). We define $\overline{N}(v)$ by including v itself in $N(v)$. We also use the set of vertices at distance at most 2 from v called the 2-neighbourhood of v and denoted $N_2(v)$ (or $\overline{N_2}(v)$ if it includes v). Finally, we write $\log n$ for the binary logarithm of n.

Time is divided into discrete synchronised time intervals (rounds) also called *slots* (following the usual terminology in wireless networks). All processes wake up and start computation in the same slot. In each slot, all processors act in parallel and either beep or listen. In addition, processors can perform an unrestricted amount of local computation in-between two slots (in effect, our algorithms require little computation).

▶ Remark. In general, nodes are active or passive. When they are active they beep or listen; in the description of algorithms we say explicitly when a node beeps meaning that a non beeping active node listens.

The time complexity, also called *slot complexity*, is the maximum number of slots needed until every node has terminated. Our algorithms are typically structured into *phases*, each of which corresponds to a small (constant or logarithmic) number of slots. In the algorithm, we specify which one is the current slot by means of a `switch` instruction with as many `case` statements as there are slots in the phase. Phases repeat until some condition holds for termination.

▶ Remark. An algorithm given in a beeping model induces an algorithm in the (synchronous) message passing model. Thus, given a problem, any lower bound on the round complexity in the message passing model also holds for slot complexity in the beeping model.

Distributed Randomised Algorithm.

A randomised (or probabilistic) algorithm is an algorithm which makes choices based on given probability distributions. A *distributed* randomised algorithm is a collection of local randomised algorithms (in our case, all identical).

A *Las Vegas* algorithm is a randomised algorithm whose running time is not deterministic, but still finite with probability 1, and that always produces a correct result. A *Monte Carlo* algorithm is a randomised algorithm whose running time is deterministic, but whose result may be incorrect with a certain probability. Put differently, Las Vegas algorithms have uncertain execution time but certain result, and Monte Carlo algorithms have certain execution time but uncertain result. Classical considerations on symmetry breaking in anonymous beeping networks (see for instance Lemma 4.1 in [1]), imply that:

▶ Remark. There is no Las Vegas (and a fortiori no deterministic) algorithm in BL which allows a node to distinguish between an execution where it is isolated and one where it has exactly one neighbour.

From this remark we deduce that there is no Las Vegas counting algorithm in BL, which advocates the use of stronger models. In what follows, we consider whichever model is the most convenient and provide Las Vegas algorithms in these models. We then present canonical emulation techniques to turn any such algorithm into a Monte Carlo one in BL.

3 Design patterns for beeping algorithms

As a warm-up, this section presents a number of design patterns which seem to occur frequently in the design of beeping algorithms. The concept of pattern refers here to reusable solutions to common problems. These patterns are then used to describe algorithms in the other sections.

Exclusive beeps.

Beeping algorithms operate in synchronous periods called *slots*, which are equivalent to the concept of rounds in message passing models. Most problems in distributed computing require some node v to take exclusive decisions at times (i.e., with respect to vertices of $\overline{N}(v)$ or $\overline{N_2}(v)$), which requires some type of symmetry breaking. In beeping networks, this goal is all the more difficult to achieve that the nodes cannot use identifiers nor even port numbers in their basic exchanges. If we assume that a node that is beeping can detect whether another node beeps simultaneously (B_{cd}), then this feature can be used to take exclusive decision if indeed it beeps alone. We call this an *exclusive beep*. Algorithm 1 illustrates an empty shell of algorithm that relies on repeated attempts to produce exclusive beeps. Most, if not all algorithms rely implicitly on this pattern as a basis.

Algorithm 1: Exclusive beeps (using B_{cd}).

```
repeat
    beep with some probability;
    if I beeped alone then
        do something exclusive;
    ...
until finished;
```

2-hop exclusive beeps.

For some problems like 2-hop colouring, 2-hop MIS, or computation of the degree (all discussed in this paper), the level of mutual exclusion offered by exclusive beeps is not sufficient and the algorithm requires that a node be the only one to beep at distance 2. Assuming collision can also be detected upon listening (L_{cd}), one can design a 2-slots pattern whereby non-beeping neighbours report if they have heard more than one beep. Hence, if a node produced an exclusive beep in the first slot, and none of its neighbours reported a collision in the second, then it knows that it has produced a *2-hop exclusive beep* (see Algorithm 2).

Algorithm 2: Two-hops exclusive beeps (using $B_{cd}L_{cd}$).

```
repeat
    switch slot do
        slot 1 // contending
            beep with some probability;
        slot 2 // detection of peripheral collision
            if several neighbours beeped in slot 1 then
                beep
        after slot 2
            if I beeped alone in slot 1 and no neighbour beeped in slot 2 then
                do something 2-hop exclusive
        ...
until finished;
```

Multi-slot phases.

The example in Algorithm 2 illustrates another common aspect of beeping algorithms, namely *multi-slot phases*. The expressivity of a single beep is rather poor, but several combined slots can achieve elaborate behavior. In Algorithm 2, one slot is devoted to contending and another to peripheral collision detection. The whole compound is then called a *phase*. Another common task is termination detection. In a *termination slot*, all nodes which have not yet performed some action beep. If the slot remain silent, then a form of local termination is detected: nodes are in a terminal state.

Adaptive probability.

As far as feasibility and expressivity are concerned, the next design pattern is not crucial. However, it plays a central role in terms of performance. *Adaptive probability* consists in adapting the probability to beep in the next phase depending on the outcome of previous phases. Typically, if a collision occurs, the probability is reduced, and if no one beeps, it is increased. Since the nodes do not know how many neighbours are contending with them, this technique proves useful in optimizing the odds of producing exclusive beeps. The values

Algorithm 3: Adaptive beeping probability (using $B_{cd}L_{cd}$).

```
Float p ← 1/2 // say
repeat
    beep with probability p;
    if I beeped alone then
        do something exclusive;
    else
        if no one beeped then
            increase p;
        else
            decrease p;
until finished;
```

given to the probabilities in Algorithm 3 are left unspecified. There are several options. In this paper, we use a doubling/halving pattern, that is, p is increased to $2p$ (up to $1/2$), and it is decreased to $p/2$ (without limit). A similar doubling/halving pattern was used in [11]. One could also increment or decrement the denominator of p as done in [3]. The consequences of choosing one over the other are not discussed here.

Collision detection.

Most algorithms in this paper use collision detection as a built-in primitive, referred to as B_{cd} for detection on beeping and L_{cd} for detection on listening. However, this feature is not always available as a primitive. An important question is the transformation of a (high-level) algorithm using B_{cd} or L_{cd} (or both) into one that works in the weakest BL model. This question is the topic of Section 5, in which we study generic mechanisms to achieve this goal. Essentially, each slot that requires collision detection can be replaced with a logarithmic number of slots (in the size of various quantities depending on the desired guarantees) where the ties are broken *w.h.p.* We provide dedicated procedures that generalise the technique used internally to one of the algorithms in [1]. Besides complexity, the price to pay is that the algorithm becomes Monte Carlo instead of Las Vegas, that is, the result is correct only probabilistically (though possibly *w.h.p.*). We present a matching lower bound showing that these procedures are essentially optimal.

4 Algorithms for basic graph problems

We now present algorithms for a number of problems, including colouring (with or without knowledge on the degree), 2-hop colouring, computation of the degree and 2-hop MIS. These algorithms are based on various combinations of the patterns presented in Section 3. All algorithms are Las Vegas, and they rely on medium to strong primitives ($B_{cd}L$ to $B_{cd}L_{cd}$ models) depending on the needs. The adaptation of these algorithms in the weakest model (BL) is discussed in Section 5. We also recall Jeavons et al.'s Las Vegas algorithm for the MIS [8] problem and discuss its relations with our 2-hop MIS algorithm.

Whenever using the adaptive probability pattern in algorithms, for generality, we stick to the terms *increase* and *decrease* (as opposed to our analyses, in which these actions are instantiated to *doubling* and *halving* the probability).

4.1 Colouring

The colouring problem consists of assigning a colour to every node in the network, such that no two neighbours have the same colour. We first consider the case that no extra information is available to the nodes. Then we consider that (an upper bound on) the maximum degree is known.

Colouring without knowledge.

Informally, the algorithm proceeds as follows (see Algorithm 4 for details). Initially, every node is uncoloured (*nil*). In every phase, each node increments a counter. Uncoloured nodes contend with each other to produce an *exclusive beep*, and when one succeeds, it takes the current value of the counter as its colour and retires. An *adaptive probability* is used to regulate the probability of beeping among uncoloured nodes. Local termination (a node and its neighbours are coloured) detection is not explicitly handled here, though we could add a *termination slot* where uncoloured nodes are the only ones to beep.

Algorithm 4: A Las Vegas colouring algorithm in $B_{cd}L$ (without knowledge).

```
Float p ← 1/2;
Integer colour ← nil;
Integer counter ← 0;
repeat
    beep with probability p;
    if I beeped alone then
        colour ← counter
    else
        if no one beeped then
            increase p;
        else
            decrease p;
        counter ← counter + 1;
until colour ≠ nil;
```

The running time of this algorithm is of $O(\log n + \Delta)$ phases *w.h.p* as well as on average (none of both imply the other trivially). Note that this is also the number of slots, since each phase consists of a *constant* number of slots. As for the number of colours, it is incremented with time, thus it is at most the same (at most, because some phases may not produce exclusive beeps).

Colouring with a bound K on the maximum degree Δ.

If a bound $K \geq \Delta$ is known, then one can obtain a better colouring using at most $K+1$ colours. The algorithm follows the same lines as Algorithm 4, i.e. a colour counter is incremented in each phase, and its current value is chosen by those nodes who produced an exclusive beep. The main difference (see Algorithm 5 for details) is that only those colours within $\{0, \ldots, K\}$ are considered and thus the counter is incremented modulo $K+1$. Conflicts of colours are avoided by keeping a phase idle if the corresponding value was already taken in the past (locally). To do so, when a node takes a colour, it *re-beeps* in a new slot called *confirmation slot* to inform its neighbours that they must remove the current colour from their list of authorized colours. Accordingly, the uncoloured will contend in a phase only if the current colour is still available (otherwise, they wait). An adaptive probability is used similarly to Algorithm 4, except that idle phases are not considered as silent (the probability is not updated in these phases).

Algorithm 5: A Las Vegas colouring algorithm in $B_{cd}L$ (knowing $K \geq \Delta$).

```
Colours = {0, ··· , K};
Float p ← 1/2;
Integer colour ← nil;
Integer counter ← 0;
repeat
    if counter ∈ Colours then
        switch slot do
            slot 1  // contending
                beep with probability p
            slot 2  // confirmation
                if I beeped alone in slot 1 then
                    colour ← counter;
                    beep;
                else
                    if no one beeped then
                        increase p;
                    else
                        decrease p;
            if someone beeped in slot 2 then
                Colours ← Colours \ {counter}
    counter ← (counter + 1)  mod (K + 1);
until colour ≠ nil;
```

Regarding performance, the only difference between this algorithm and Algorithm 4 is that a growing number of phases are idle in each neighbourhood, inflicting a slow down to the algorithm. Managing the dependencies here proved more difficult and we "only" managed to prove that the number of phases is $O\left(K(\log n + \log^2 K)\right)$ *w.h.p.* However, the algorithm is believed to be faster.

4.2 2-hop colouring

A 2-hop colouring of a graph G is a colouring such that any two nodes at distance ≤ 2 have different colours. In other words, it is a colouring of the square of G, the graph where an edge exists between nodes which are neighbours in G or share a common neighbour in G.

2-hop colouring without knowledge.

A similar strategy is used as in Algorithm 4 (colouring), except that exclusive beeps are replaced with *2-hop exclusive beeps*. Whenever a node produces such a beep, it takes the current value of the counter as colour. Since no other node has beeped within distance 2, the colouring is legal. Contrary to the 1-hop colouring, the collaboration of a node remains crucial even after it becomes coloured. Indeed, this node must keep on reporting peripheral collisions to its neighbours. As a result, instead of retiring from computation, coloured nodes keep on listening until all of their neighbours are coloured, which is detected using an extra *termination slot*. Details are given in Algorithm 6. Four slots are used in total, the first two being devoted to the management of 2-hop exclusive beeps (see Section 3 for details). The third slot manages a (2-hop) adaptive probability based on beeps heard at distance one (slot 1) or at distance two (slot 3 itself). Finally, slot 4 is the termination slot.

Algorithm 6: A Las Vegas 2-hop-colouring algorithm in $B_{cd}L_{cd}$ (without knowledge).

```
Float p ← 1/2;
Integer colour ← nil;
Integer counter ← 0;
repeat
    switch slot do
        slot 1 // contending slot
            if colour = nil then
                beep with probability p;
        slot 2 // peripheral collision detection (and consequences)
            if several neighbours beeped in slot 1 then
                beep
            if I beeped alone in slot 1 and heard no beep in slot 2 then
                colour ← counter
        slot 3 // adaptive probability
            if someone beeped in slot 1 then
                beep
            if colour = nil then
                if no beep heard in slot 1 nor 3 then
                    increase p
                else
                    decrease p
        slot 4 // termination slot
            if colour = nil then
                beep
    counter ← counter + 1
until no beep heard in slot 4;
```

Once we realize that the execution produced here is the same as what Algorithm 4 would produce in the square of G, analysis of this algorithm is straightforward. The only difference is that the maximal number of contenders of a node becomes Δ^2 instead of Δ. Thus Algorithm 6 takes $O(\log n + \Delta^2)$ phases (and slots) *w.h.p.*, and the number of colours cannot exceed the same value.

With a bound K on the maximum degree Δ.

The same idea can be applied as in the 1-hop variant, *i.e.*, taking colours between 0 and $K^2 + 1$ (instead of $K + 1$) and incrementing the counter accordingly (mod $K^2 + 1$). As a result, at most $K^2 + 1$ colours are used, with time complexity $O(K^2(\log n + \log^2 K))$ *w.h.p.*

4.3 Degree computation

Let us recall that 2-hop exclusive beeps allow a node v to perform an exclusive action within a radius of distance 2. This feature was used in Section 4.2 to assign unique colours. At it turns out, the pattern is very versatile and it can be used to count the degree of a node as well. The strategy consists in replacing the colour-related action in slot 2 (second `if-then` block) by an action aiming at having v counted in the degree of its neighbours (then v stops contending and keeps on reporting collisions, as before). Precisely, a new confirmation slot is inserted wherein v re-beeps if indeed it produced a 2-hop exclusive beep. Upon hearing the confirmation beep, all of v's neighbours increment a local counter that eventually amounts to their degree. Termination proceeds in the same way as for the 2-hop-colouring algorithm (i.e. uncounted nodes beep in a termination slot).

Up to a constant factor which accounts for the additional confirmation slot in each phase, the running time of this algorithm is again $O(\log n + \Delta^2)$ *w.h.p.*

4.4 Jeavons et al.'s Las Vegas Algorithm for the MIS in $B_{cd}L$

We recall here Jeavons et al.'s Las Vegas Algorithm for the MIS [8]. This algorithm uses an *adaptive probability* to maximize the frequency of exclusive beeps (with a doubling/halving pattern for p, starting at 1/2). If a node v produces an exclusive beep, it enters the MIS (by the end of the first slot), then it uses a confirmation slot to inform its neighbours, all of which terminate together with v. Since the whole neighbourhood shuts down at once, the algorithm progresses faster than, for instance, the basic colouring algorithm discussed above. This algorithm was already proven by Jeavons et al. to terminate within $O(\log n)$ slots with a huge constant factor (larger than e^{25}).

4.5 Computing a 2-hop MIS

In this problem, we must select a set of nodes (the MIS) such that no pair of selected nodes are within distance 2 and no node can be added further to the set. This algorithm is a combination of those of other 2-hop algorithms seen above, and Jeavons et al's MIS algorithm. That is, the same structure of algorithm is used as for 2-hop colouring or degree computation, except that whenever a node produces a 2-hop exclusive beep, it enters the MIS and informs its neighbours (using the confirmation slot) that they will not be in the MIS. This algorithm takes the same number of *phases* than what the (1-hop) MIS algorithm would produce in the square of the graph, that is, $O(\log n)$ *w.h.p.*. The number of slots is higher due to using additional slots for managing 2-hop exclusive beeps, but it remains within a (small) constant factor. Interestingly, our analysis of this algorithm improves much over that of [8], taking the huge constant down to 76 (i.e., making the algorithm practical).

5 Collision detection and emulation techniques

In Section 4, we have considered collision detection as a built-in primitive. Depending on the algorithms, we assumed that collision detection was possible while beeping (B_{cd}) or while listening (L_{cd}). This assumption is convenient because it allows one to design *Las Vegas* algorithms for all the considered problems. Unfortunately, we know since [1] that no Las Vegas algorithms can be designed for most problems without collision detection, that is, in the BL model. One has to turn to Monte Carlo instead, which means that the result is correct only with some probability (possibly *w.h.p.*). In this section, we investigate the cost of building a probabilistic collision detection primitive in the BL model, inspired by a technique from [1]. Then we adapt it into two generic emulation procedures, one for detecting collision while beeping, the other while listening. These procedures can then be used to translate any Las Vegas algorithml in $B_{cd}L$, BL_{cd}, or $B_{cd}L_{cd}$, into a Monte Carlo algorithm in BL. The cost is a logarithmic slowdown of the execution, which we prove is essentially optimal (for sufficiently large n).

5.1 Collision detection

The impossibility for a node in BL to distinguish between begin alone or having neighbours has strong implications. For instance, in the colouring problem, it means that two neighbours could possibly end up with the same colour. In the MIS problem, two neighbours could enter the MIS. In fact, there is no guarantee on the correctness of basic patterns like exclusive beeps or 2-hop exclusive beeps, which are at the basis of most (if not all) Las Vegas algorithms.

We present a (Monte Carlo) algorithm for detecting collisions in BL. This procedure generalises the technique used in Algorithm 3 of [1], which consists of replacing each slot that requires collision detection in the original model, with several BL slots in which symmetries are probabilistically broken. Of course, the more slots, the more reliable the detection.

The algorithm.

Each slot that requires collision detection (B_{cd} or L_{cd}) is replaced with a number of *sub-phases*, each consisting of two BL slots. For instance, if a node wishes to beep with collision detection in the original algorithm, it will choose one of the two slots (*u.a.r.*) in each of the sub-phases and will beep in that slot (listen in the other). If it hears a beep while listening in the other slot, then an internal collision is detected. Similarly, if a node wishes to listen with collision detection in the original algorithm, it will listen in both slots of each sub-phase. A peripheral collision is detected if a beep is heard in both slots of a same sub-phase. The procedure is detailed by Algorithm 7, where k is the number of sub-phases used.

False positives never happen, but real collisions might still go unnoticed, with probability inversely related to k. We are interested in determining how large k should be to guarantee that a given node detects a collision in its neighbourhood with a given probability. The stronger question asks how many sub-phases are required to guarantee that *none* of the nodes fails to detect a collision.

▶ **Lemma 1.** *Let v be a node. If a collision occurs in the neighbourhood of v, then v detects it in $O\left(\log(\frac{1}{\epsilon})\right)$ sub-phases (slots) with probability at least $1-\epsilon$, and in $O\left(\log n\right)$ sub-phases (slots) with probability $1-o\left(\frac{1}{n^2}\right)$.*

Algorithm 7: Collision detection algorithm in BL (with parameter k)

```
Boolean collision ← false;
Integer i ← 0;
while i < k do
    if v wishes to beep then
        Flip a coin;
        if heads then
            beep in slot 1;
            listen in slot 2;
        else
            listen in slot 1;
            beep in slot 2;
        if another beep was heard then
            collision ← true
    else
        listen in both slots;
        if beeps are heard in both slots then
            collision ← true;
    i ← i + 1;
return collision;
```

Proof. Assume a collision occurs between some nodes u_1 and u_2 in the neighbourhood of v (one of them being possibly v itself). It is detected if u_1 and u_2 choose a different slot in at least one of the k sub-phases. The probability that this does not happen is $\left(\frac{1}{2}\right)^k$. This probability is less than ϵ (resp. $o\left(\frac{1}{n^2}\right)$) for any $k \geq \log(\frac{1}{\epsilon})$ (resp. $2\log(n)$). Observe that if collisions occur between more than two nodes in the neighbourhood of v, this cannot decrease the odds of a successful detection (to the contrary, the odds can only increase). ◀

▶ **Corollary 2.** *Let G be a graph. If collisions occur in the neighbourhood of an arbitrary number of nodes, then all of them detect collision after at most $O\left(\log(\frac{n}{\epsilon})\right)$ sub-phases (slots) with probability at least $1-\epsilon$, and after at most $O\left(\log n\right)$ sub-phases (slots) w.h.p.*

Proof. Assume collisions occur in G and let T denote the number of sub-phases before all concerned nodes detect collision. Clearly $T = \max\{T_v \mid v \in V\}$, where T_v is the time it takes to any node v to decide collision. By the same argument as in the proof of Lemma 1, together with union bound, it holds that

$$\mathbb{P}r\left(T > \log\left(\frac{n}{\epsilon}\right)\right) \leq n \times \mathbb{P}r\left(T_v > \log\left(\frac{n}{\epsilon}\right)\right) \quad (1)$$

$$= n \times \frac{1}{2^{\log(\frac{n}{\epsilon})}} = \epsilon \quad (2)$$

which proves the first claim. The same argument, combined with the second claim of Lemma 1 proves the second claim. ◀

5.2 Emulation procedures

Based on this tie-breaking mechanism, we define two probabilistic emulation procedures whose purpose is to replace beep or listen instructions with collision detection in BL.

Both are Monte Carlo in the sense that detection is only guaranteed with some probability. The first procedure, `EmulateB`$_{cd}$`inBL()`, is given by Algorithm 8 and the second, `EmulateL`$_{cd}$`inBL()`, by Algorithm 9. Both procedures are parametrized by an integer $k > 1$, which accounts for the number of sub-phases that are used in each invocation of the procedure (k controls the error bound). They return `true` if a collision has been detected, `false` otherwise.

Before the execution each vertex generates a sequence s of k random bits ($u.a.r.$) which will be the ones used in each sub-phase. The reason why this is made once at the beginning rather than in each invocation is a technicality that relates to preventing an additional union bound in the analysis (more k would be needed to guarantee that *each* invocation is successful if the numbers are drawn every time).

Algorithm 8: A Procedure to emulate a B_{cd} in the BL model.

```
Procedure EmulateL_cd inBL(in : Integer k, Array s; out : Boolean collision)
Boolean collision ← false;
Integeri ← 0;
repeat
    if s[i] then beep in slot 1; listen in slot 2;
    else listen in slot 1; beep in slot 2;
    if another beep was heard then collision ← true;
    i ← i + 1
until i = k;
End Procedure
```

Algorithm 9: A Procedure to emulate a L_{cd} in the BL model.

```
Procedure EmulateL_cd inBL(in : Integer k; out : Boolean beep, Boolean collision)
Boolean beep ← false;
Boolean collision ← false;
Integer i ← 0;
repeat
    switch slot do
        slots 1 and 2
            listen
        end of phase:
            if a beep was heard in any slot then
                beep ← true
            if a beep was heard in both slots then
                collision ← true
    i ← i + 1
until i = k;
End Procedure
```

Hence, the value of k depends on the bound we require on the probability of error, a straightforward adaptation of the above analysis gives us the values of Lemma 3.

▶ **Lemma 3.** *For any $\varepsilon > 0$, and any $n > 0$:*

1. *if $k = \lceil \log\left(\frac{1}{\varepsilon}\right) \rceil$, the procedures are correct for a given node with probability $1 - \varepsilon$*

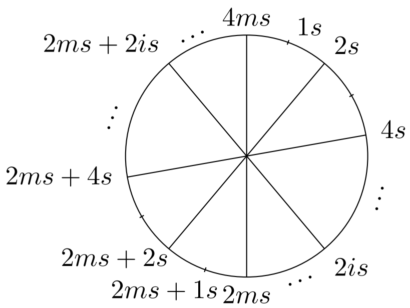

Figure 1 The wheel gadget used in the proof of optimality for emulation.

2. *if* $k = \lceil \log\left(\frac{n}{\varepsilon}\right) \rceil$*, the procedures are correct for any node with probability* $1 - \varepsilon$
3. *if* $k = \lceil 2\log(n) \rceil$*, the procedures are correct for any node w.h.p.*

Observe that in general, the size of the network n is not known to the nodes, which is an obstacle to achieving the second and third types of guarantees. However, it is reasonable in practice to assume that the nodes know an *upper bound* on n, e.g., when a network of wireless sensors is deployed. The upper bound may even be loose without much consequence: so long as it is polynomial in n, the slowdown factor remains of the same order.

Using the procedures.

In the listings of our algorithms (see Section 4), listen instructions are implicit. By default, a node listens if it does not beep. Emulation procedures should be used explicitly for both *beep* and *listen* primitives, in order for the nodes to remain synchronized (since each of them takes logarithmically many rounds to be carried out). Therefore, whenever a node calls `EmulateB`$_{cd}$`inBL` or `EmulateL`$_{cd}$`inBL`, the other nodes should call one of these or wait the corresponding amount of time. Likewise, the procedures should not be interrupted even after a collision with a given neighbor is detected, to preserve synchrony with other neighbours or farther nodes.

5.3 Optimality of the emulation

In this section we prove that the emulation procedures presented in Section 5.2 are essentially optimal (i.e. asymptotically and up to a constant factor), namely, we prove a $\Omega(\log n)$ lower bound on the number of slots required to detect collision in some graphs called *wheels*. A (m, s)-wheel, illustrated in Figure 1, is a graph $W = (V, E)$ such that $V = u_1, \ldots u_{4ms}$, the edges E are all the (u_{i-1}, u_i) (arithmetic modulo $4ms$) plus m spokes, that is edges $(u_{is}, u_{(i+2m)s})$ $(1 \leq i \leq 2m)$, where the wheel can be odd (all spokes with i odd) or even (all spokes with i even). The even and odd (m, s)-wheels are isomorphic. We consider only situations in which all vertices u_{is} are in the same state, a state in which they wish to beep and all other vertices are in the same internal state, a state in which they do not wish to beep. Thus vertices at the ends of spokes and no others must conclude that there is a collision. The slot complexity of any algorithm which detects collision in such a graph with high probability is to be $\Omega(\log n)$. Due to space limitations, the full proofs are relegated to Appendix B. We provide, however, a minimal sentence of insight for each.

Considering a computation of a collision detecting algorithm on a wheel, we define, for any $t > 0$, b_t^i as the signal (beep or not) from u_i to all its neighbours at time t, and, for any

$t \geq 0$, B_t^i the sequence $b_1^i \cdots b_t^i$. Then, we define the event E_t for a spoke $u_{is}, u_{(i+2m)s}$ as follows:

$$E_t = \left\{ (B_t^{is} = B_t^{(i+2m)s}) \wedge (B_t^{is+1} = B_t^{(i+2m)s+1}) \wedge (B_t^{is-1} = B_t^{(i+2m)s-1}) \right\}.$$

▶ **Lemma 4.** *For any t ($0 \leq t < s$), it holds that $\mathbb{P}r\,(E_t) \geq 2^{-3t}$.*

The proof proceeds by induction on t, with base case $t = 0$. (Full proof in Appendix B.)

If E_t holds for the spoke $(u_{is}, u_{(i+2m)s})$, we say that the spoke fails to break symmetry within time t. This happens with probability at least 2^{-3t} and, if it happens, the existence of the spoke has had no influence on the computation up to time t. In particular, whenever u_{is} beeped, $u_{(i+2m)s}$ also beeped and so neither has ever heard the other beep.

▶ **Theorem 5.** *For any Monte Carlo algorithm $\mathcal{A}$ which detects collision in W, if $\mathcal{A}$ halts in less than $\log_2 n/4$ rounds with probability greater than $3/4$ then for some situations in some wheels, $\mathcal{A}$ gives incorrect results for some vertices with probability greater than $1/4$.*

The proof proceeds using the wheel gadget of Figure 1 and Lemma 4 to characterize the rate at which the symmetry induced by the spokes can be broken. (Full proof in Appendix B.)

▶ **Corollary 6.** *The complexity of a Monte Carlo algorithm which detects collision with high probability in the BL model is $\Omega(\log n)$.*

References

1 Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, and F. Kuhn. Beeping a maximal independent set. *Distributed Computing*, 26(4):195–208, 2013.

2 K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.

3 A. Casteigts, Y. Métivier, J. Michael Robson, and A. Zemmari. Counting in one-hop beeping networks. *CoRR*, abs/1605.09516, 2016.

4 J. Collier, N. Monk, P. Maini, and J. Lewis. Pattern formation by lateral inhibition with feedback: a mathematical model of delta-notch intercellular signalling. *Journal of Theoretical Biology*, 183(4):429–446, 1996.

5 A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *DISC*, pages 148–162, 2010.

6 S. Gilbert and C. Newport. The computational power of beeps. In *Proc. of 29th International Symposium on Distributed Computing (DISC)*, 2015.

7 B. Huang and Th. Moscibroda. Conflict resolution and membership problem in beeping channels. In *DISC*, pages 314–328, 2013.

8 P. Jeavons, A. Scott, and L. Xu. Feedback from nature: simple randomised distributed algorithms for maximal independent set selection and greedy colouring. *Distributed Computing*, DOI 10.1007/s00446-016-0269-8, 2016.

9 S. Navlakha and Z. Bar-Joseph. Distributed information processing in biological and computational systems. *Commun. ACM*, 58(1):94–102, 2015.

10 J. Schneider and R. Wattenhofer. What is the use of collision detection (in wireless networks)? In *DISC*, pages 133–147, 2010.

11 A. Scott, P. Jeavons, and L. Xu. Feedback from nature: an optimal distributed algorithm for maximal independent set selection. In *PODC*, pages 147–156, 2013.

A Complexity analysis

This appendix section provides the analysis and proofs of the running time complexity of the algorithms presented in Section 4.

A.1 Colouring algorithm without knowledge

Informally the execution progresses as follows. There is a first period of adjustment in which the probabilities will converge towards "good values". Then the probability that an exclusive beep is produced in a given phase in a given neighbourhood remains bounded in some ways. Loosely speaking, the final bound is essentially obtained by a repetition of the corresponding periods Δ times. More precisely, we prove the following theorem:

▶ **Theorem 7.** *There are constants α, β and γ such that for any graph $G = (V, E)$ of n vertices and maximum degree Δ, the number of phases of Algorithm 4 to colour all the nodes in G is:*

1. *less than $\alpha(\Delta + \log n)$ with probability $1 - o\left(n^{-1}\right)$,*
2. *less than $\beta(\Delta + \log n)$ on average,*
3. *less than $\gamma(\Delta + \log n)$ with probability $1 - o\left(n^{-c}\right)$, for any $c > 1$. (This result is stronger than 1, but the proof is more difficult, which is why we keep both.)*

A.1.1 Local Average Time Complexity.

First, we give an overview. We define p_v as the probability that vertex v claims the colour in a given round and q_v as the sum of p over all neighbours of v which we will call u_i $(1 \le i \le d)$ where d is v's degree in the residual graph (taken as 0 if v has been eliminated from the graph).

We define a measure M of the distance from a given situation to the goal where $p = 1/2,\ q_v \le 1/2,\ d = 0$ as follows: $M = -\log(p) + f(q) + 10d$ where f is the function defined as follows:

- $f(x) = 4x$ if $x \le 1$,
- for $x > 1$, f is the piecewise linear approximation to $2\log_2 4x$ where$f$ is interpolated linearly between $f(2^i) = 2i + 4$ and $f(2^{i+1}) = 2i + 6$.

We note the following properties of f which will be used in what follows:

- $f(x)$ is continuous for $x > 0$,
- except at powers of 2, f is differentiable with derivative $\le$ 4,
- $f(x) - f(x/2) = 2$ for $x \ge 1$,
- $f(x) - f(x/2) = 2x$ for $x \le 1$.

We show that in any round, the mean decrease in M is at least 1. Then after a number of rounds equal to the initial M $(\le 1 + 2\log(2d) + 10d)$, M is reduced on average to 0 unless the algorithm has already terminated at v and after $O(\log n)$ further rounds, the algorithm has terminated at v with probability $o(n^{-2})$ and so it has terminated everywhere with probability $o(n^{-1})$.

Intuitively, we expect both p and q to decrease initially until $q < 1/2$, after which p will re-ascend until it is at least close to $1/2$ and then d will start to descend.

We actually analyse the variation in a random variable M' which dominates the r.v. M. The r.v. M' is initially equal to M but its changes may be slightly different from those of M in the following ways:

- if the degree of v decreases by more than 1, the change in M' only includes -10 in total for the change in d rather than -10 for each neighbour removed;
- if v takes a colour, M' is decreased by just 10 whatever the values of p at v and its neighbours;
- in a round where v is already coloured, M' is decreased by 1.

This ensures that:

- if the algorithm has not terminated at v, $M' \geq M$, meaning that M' dominates M;
- if $M' \leq 0$, the algorithm has terminated at v;
- M' decreases at each round by a value in $[-3 \dots 11]$ (since, $\log p$ can only change by ± 1, $f(q)$ can only change by up to ± 2, and if d decreases because a neighbour u takes the colour, u beeped in the round and so p has halved).

A.1.2 Available Neighbours.

In a given round any u_i has a well defined probability a_i of being *available* that is able to take the current colour since no neighbour claims it. These probabilities are far from being independent. We will argue that, except for cases where p_{dec} (the probability of a decrease in d) is at least $2/5$, the average decrease in M' is always minimised when all $a_i = 0$.

Consider a situation where $p_{dec} < 2/5$ and some $a_i > 0$. We can decrease a_i to 0 with no change to the other a_j by adding an infinite number of vertices adjacent to u_i but to no other vertex in $\overline{N}(v)$. This will change the average increase/decrease in q and d; q_i will be halved instead of doubled with probability a_i, decreasing q by $3q_i/2$ and so decreasing $f(q)$ by at most $6a_iq_i$ on average. The probability of a decrease in d is decreased by a_i times the probability that u_i claims and no other u_j takes the colour. This last probability is the product of q_i and the conditional probability that no other u_j takes the colour given that u_i claims. But, since the probabilities of u_i claiming and of some u_j $(j \neq i)$ taking the colour are negatively correlated or independent, this conditional probability is at most the unconditional probability that no u_j $(j \neq i)$ takes the colour and so greater than the probability that no u_j takes the colour, namely $1 - p_{dec} \geq 3/5$, giving an average decrease in d (respectively M') reduced by more than $3a_iq_i/5$ (resp. $6a_iq_i$).

Thus the decrease in the measure is decreased more by the d component than it is increased by the change in q.

Repeating this process at most d times we arrive at a situation with a smaller mean decrease in M' than the initial one and either $p_{dec} \geq 2/5$ or all $a_i = 0$ so, to lower-bound the decrease in M', we need only consider such situations.

A.1.3 The mean decrease in M'.

We consider cases depending on the value q. First note that if $p_{dec} \geq 2/5$, the mean decrease in M' is at least $10(2/5) - 2 - 1 = 1$. So in the other cases we suppose that all $a_i = 0$ so that q is halved.

In the case where $q < 1$, we need to consider what happens when no u_i claims the colour. If $p < 1/2$ this is that p increases, decreasing M' by 1; if $p = 1/2$ it is that, with probability $1/2$, v takes the colour so that d decreases by 1, decreasing M' by 10 so that on average M' decreases by 5. Accordingly we suppose that the former happens.

- $q \geq 1$: q decreases to $q/2$, reducing $f(q)$ by 2 and $\log p$ can decrease by at most 1 so that M' is decreased by at least $2 - 1 = 1$.

- $q < 1$: q decreases to $q/2$, decreasing $f(q)$ by $2q$ while p doubles with probability at least $1 - q$ (and halves with probability at most q). This gives a mean decrease in M' of at least $2q + (1 - 2q) = 1$.

A.1.4 Time Complexity *w.h.p.*.

We define the sequence of r.v.'s $(M_k)_{0\leq k\leq t}$ as follows $M'_0 = M_0$ and for any $k \geq 1$, M_k is the value of M' after time k. We also define the sequence $(G_k)_{0\leq k\leq t}$ as the sequence of residual graphs, i.e., $G_0 = G$ and G_{k+1} is the graph obtained from G_k after round $k+1$ (each vertex which succeeds in beeping alone is removed from the graph).

Then for any $k \geq 1$:

$$\mathbb{E}\left(M_k \mid G_1, G_2, \cdots, G_{k-1}\right) \leq M_{k-1} - 1. \tag{3}$$

Hence, $(M_k)_{k\geq 0}$ is a super-martingale with respect to $(G_k)_{k\geq 0}$.

We define the r.v. $D_k = M_k - M_{k-1}$ for any $k \geq 1$ and we denote $\mu = \mathbb{E}\left(D_k\right)$. We also introduce the r.v.:

$$D''_k = -\frac{4}{\mu - 3} D_k + \frac{3\mu + 3}{\mu - 3}.$$

Then, it is easy to see that $\mathbb{E}\left(D''_k\right) = -1$ and $\mathbb{P}r\left(-11 \leq D''_k \leq 3\right) = 1$.

Now, define the r.v. $(M''_k)_{k\geq 0}$ as follows: $M''_0 = M_0$ and for any $k \geq 1$, $M''_k = M''_{k-1} + D''_k + 1$. Then $(M''_k)_{k\geq 0}$ is a martingale with respect to $(G_k)_{k\geq 0}$.

We apply Theorem 18 of [2] to our martingale M''_t with expectation M_0. Since the increments $(D''_k + 1)_{k\geq 0}$ are in $[-10..4]$ and have mean 0, their variance is upper bounded by the case of a distribution with values -10 and 4 with probabilities 2/7 and 5/7 respectively, giving variance of 40 and maximum discrepancy from the mean of 10. Applying the theorem with $t = 2M_0 + 174 \ln n$ and $\lambda = t - M_0$, we see that the probability that $M_t \geq 0$ is less than $\mathbb{P}r\left(M''_t \geq t\right)$ which is at most:

$$e^{\left(-\lambda^2/2(40t+10\lambda/3)\right)},$$

and we claim that this is $o(n^-2)$. This is because $\lambda^2/2(40t + 10\lambda/3) >> 2 \ln n$, i.e. $\lambda^2 >> 4 \ln n(40t + 10\lambda/3)$.
(Proof of this claim: $\lambda^2/13 >> 4 \ln n(10\lambda/3)$ because $\lambda >> 520 \ln n/3$;
$12\lambda^2/13 > 12(t^2 - 2M_0 t)/13 = 12t(t - 2M_0)/13 = 12t(174 \ln n)/13 >> 160t \ln n$. Adding these two gives the claim.)

Then taking $\alpha = 174$, this proves the first claim of Theorem 7.

A.1.5 Average Time Complexity.

We first prove the following lemma:

▶ **Lemma 8.** *Let v be any vertex in G, and $t > 0$. We have:*

$$\mathbb{P}r\left(v \text{ is not coloured at time } t\right) \leq e^{-\frac{3}{260}(t-2M_0)}.$$

Proof. Let T_v denote the time before v gets coloured. Then, by discussions above, taking $\lambda = t - M_0$ in Theorem 18 of [2]:

$$\mathbb{P}r\left(T_v - M_0 > \lambda\right) \leq e^{-\frac{\lambda^2}{2(40t+\frac{10}{3}\lambda)}}.$$

On the other hand, $\lambda \geq t - 2M_0$ and hence, a simple computation yields:

$$\lambda^2 > \frac{3}{13}(t - 2M_0)(4t + \frac{\lambda}{3}),$$

which proves the lemma. ◀

Back to Theorem 7. Let T denote the time before *all* the vertices in the graph are coloured. Then:

$$\mathbb{E}(T) = \sum_{t \geq 1} \mathbb{P}r(T \geq t).$$

Now, let $t_0 = 2M_0 + \frac{260}{3} \ln n$ then:

$$\begin{aligned}\mathbb{E}(T) &= \sum_{t=1}^{t_0} \mathbb{P}r(T \geq t) + \sum_{t > t_0} \mathbb{P}r(T > t) \\ &\leq 2M_0 + \frac{260}{3} \ln n + \sum_{t > t_0} \mathbb{P}r(T > t).\end{aligned}$$

On the other hand, for any $t > 0$:

$$\mathbb{P}r(T > t) \leq \sum_{v \in V} \mathbb{P}r(T_v > t) \leq n e^{-\frac{3}{260}(t - 2M_0)}.$$

Yielding:

$$\begin{aligned}\mathbb{E}(T) &\leq 2M_0 + \frac{260}{3} \ln n + n \sum_{t > t_0} e^{-\frac{3}{260}(t - 2M_0)} \\ &= 2M_0 + \frac{260}{3} \ln n + n \sum_{t > 0} e^{-\frac{3}{260}(t + t_0 - 2M_0)} \\ &= 2M_0 + \frac{260}{3} \ln n + \sum_{t > 0} e^{-\frac{3t}{260}} \\ &= 2M_0 + \frac{260}{3} \ln n + \frac{1}{1 - e^{-\frac{3}{260}}}.\end{aligned}$$

Taking $\beta = 87$, this proves the second claim of Theorem 7.

A.1.6 Time Complexity With Very High Probability.

To prove the last claim, let $c > 1$ and take $t = \frac{260}{3}(c+1)\ln n + 2M_0$ in Lemma 8, this gives:

$$\mathbb{P}r\left(T_v > \frac{260}{3}(c+1)\ln n + 2M_0\right) \leq \frac{1}{n^{c+1}} = o\left(\frac{1}{n^c}\right).$$

Thus, taking $\gamma = 87$ proves the last claim of Theorem 7.

A.2 Colouring algorithm knowing $K \geq \Delta$.

▶ Remark. We can consider the *modified* colouring algorithm, deduced from Algorithm 5, defined in the following way. By a *cycle* we mean K rounds considering the K colours. Now, every vertex uses the value of $|Colours|$ at the start of each cycle to decide the beeping probability it uses throughout this cycle.

We have the following theorem:

▶ **Theorem 9.** *Let G be a graph of size n, let K be an upper bound on the maximum degree of G. Algorithm 5 computes a $K+1$ colouring of G in at most $O\left(K(\log n + \log^2 K)\right)$ slots w.h.p.*

Proof. We consider the Colouring algorithm in which every node has the same upper bound K on the maximum degree. We consider both the *basic* algorithm in which v uses the current value of $|Colours|$ to decide its beeping probability and also the *modified* algorithm in which it uses the value at the start of the current cycle. We recall that by a *cycle* we mean K phases considering the $|Colours|$ colours.

We consider P_k the probability that vertex v survives uncoloured over k cycles.

In what follows:

- i ranges over $1..k$,
- c ranges over the C_i colours possible for v at the start of cycle i,
- u ranges over the neighbours of v still uncoloured at the start of cycle i,
- $p_u(i,c)$ is the probability that u beeps at colour c in cycle i.

First we consider the probability p that v survives uncoloured in a single phase using a colour $c \in colours(v)$. Then:

$$\begin{aligned} p &= \mathbb{P}r\left(v \text{ does not beep at colour } c \text{ in cycle } i\right) \\ &+ \mathbb{P}r\left(v \text{ does beep and some neighbour } u \text{ also beeps}\right), \end{aligned}$$

but $\mathbb{P}r\left(v \text{ does beep}\right) \geq 1/2C_i$ and the beeping probabilities of v and its neighbours are independent giving:

$$\begin{aligned} p &\leq (1-1/2C_i) + \mathbb{P}r\left(\text{some neighbour beeps}\right)/2C_i \\ &= (1-1/2C_i)\left(1+\mathbb{P}r\left(\text{some neighbour beeps}\right)/(2C_i-1)\right) \\ &\leq (1-1/2C_i)\left(1+\sum_u p_u(i,c)/(2C_i-1)\right). \end{aligned}$$

After the first phase, $p_u(i,c)$ and C_i are random variables dependent on what has happened so far, and we consider the tree of all possible executions up to k cycles, where each tree node has its own value of p. It is easily shown by induction that P_k is upper bounded by the maximum over all paths in this tree of the product of the values of p along the path. We fix a path which gives this maximum and bound the product for this path. We have the probability of surviving cycle $i \leq (exp(-1/2) * \prod_c (1+\sum_u p_u(i,c)/(2C_i-1))) \leq exp(-1/2+\sum_c \sum_u p_u(i,c)/(2C_i-1))$ and so $P_k \leq exp(-k/2+\sum_i \sum_c \sum_u p_u(i,c)/(2C_i-1))$.

We will give an upper bound on $\sum_i \sum_c \sum_u p_u(i,c)/(2C_i-1)$.

We number v's neighbours in the initial graph from 1 to $deg(v)$ in decreasing order of their *lifetime*, that is the number of phases in which they remain uncoloured.

Thus as long as u_j is not coloured the degree of v in the residual graph is at least j and so $|colours(v)| > j$.

We write $p_u(i,c)$ as $base+\delta$ where $base = 1/2C_i$ and δ is what has been added as a result of $colours(u)$ being decreased before colour c and we will bound $\sum_i \sum_u \sum_c base/(2C_i-1)$ and $\sum_u \sum_i \sum_c \delta/(2C_i-1)$ separately.

Firstly *base*: in cycle i, v has C_i colours available and so has less than C_i neighbours; each neighbour u has $\sum_c base \leq 1/2$, giving, for this cycle, $\sum_u \sum_c base/(2C_i - 1) \leq 1/4$ so that $\sum_i \sum_u \sum_c base/(2C_i - 1) \leq k/4$.

Secondly δ: For the modified algorithm $\delta = 0$. In the basic algorithm, a node u_j initially has K colours available and when (if) this number decreases from l to $l-1$, $p_u(i, c)$ increases from $1/2l$ to $1/2(l-1)$ and this increase of $1/2l(l-1)$ affects δ only for the, at most, $l-1$ colours still to be considered in this cycle so that $\sum_c \delta$ for a cycle is at most $\sum_l 1/2l$, the sum being taken over those l for which the number of colours is reduced from l. This gives an upper bound on $\sum_i \sum_c \delta/(2C_i - 1)$ of $\log K/2(2j + 1)$ since $C_i > j$ and so $\sum_u \sum_i \sum_c \delta/(2C_i - 1) < \sum_j \log K/2(2j + 1) < \log^2 K/4$.

Hence, by standard arguments, after $k = O(\log n + \log^2 K)$ cycles for the basic algorithm or $O(\log n)$ cycles for the modified algorithm, v has probability $o(1/n^2)$ of remaining uncoloured and the graph has probability $o(1/n)$ of having any uncoloured node.

◀

A.3 2-hop colouring

To calculate a 2-hop colouring of a graph G, we need to calculate a colouring of the "square" of G, that is the graph with the same vertices as G and an edge between any pair v and w of vertices which either are neighbours in G or have a common neighbour in G. In this context, Theorem 7 becomes:

▶ **Theorem 10.** *There are constants α, β and γ such that for any graph $G = (V, E)$ of n vertices and maximum degree Δ, the number of phases of Algorithm 6 to calculate a 2-hop colouring in G is:*

1. *less than $\alpha(\Delta^2 + \log n)$ with probability $1 - o\left(n^{-1}\right)$,*
2. *less than $\beta(\Delta^2 + \log n)$ on average,*
3. *less than $\gamma(\Delta^2 + \log n)$ with probability $1 - o\left(n^{-c}\right)$, for any $c > 1$.*

▶ Remark. The same transformation can be done starting from Algorithm 5 when we know an upper bound of the maximum degree.

A.4 Analysis of Jeavons et al.'s Las Vegas Algorithm for the MIS in $B_{cd}L$

In [8], Jeavons et al. give and analyse a Las Vegas beeping algorithm to compute a MIS in the model $B_{cd}L$. They prove that for any graph G with n vertices, their algorithm terminates in at most $K_0 \log n$ phases, with probability at least $1 - o(n^{-1})$ and $K_0 \geq e^{25}$.

The starting point of our work is the observation made by Scott et al. at the end of Section 4 in [11]: "Our simulations show that in practice the constants are rather lower". We verify this observation by proving that the number of phases taken by the Jeavons et al. algorithm on any graph with n vertices is at most $76 \log n$ *w.h.p.*

We first introduce some notation that we will use in this section.

If a neighbour of v beeps (in a slot), we say that v is "inhibited" (in that slot). For any vertex v, we define the following sum:

$$q_v = \sum_{u \in N(v)} p_u.$$

We also note $q_v^* = \max\{q_v, 1/5\}$ and finally $t_0 = 3\log(5q_v^*) - 2\log p_v$. We omit the subscript v where there is no risk of ambiguity.

We finally write $l(q)$ for $\log(5\max\{q, 1/5\})$, that is $l(q) = \max\{\log(5q), 0\}$.
Then, we have the following theorem:

▶ **Theorem 11.** *For any $t \geq 0$ and for any vertex v, its probability of remaining active after the next t phases is at most α^{t_0-t} for the constant $\alpha = 2^{1/36} \approx 1.01944$.*

Proof. Note that $\alpha^{3\log q} = q^{3\log\alpha} = q^{1/12}$. The proof will be by induction on t. We have $t_0 \geq 2$, so that if $t = 0$, $\alpha^{t_0-t} > 1$ and the claim is trivially true.

Let $t > 0$. After one phase which does not add v or a neighbour to the MIS we have by induction that the probability of remaining active for the following $t-1$ phases is at most $\alpha^{t'_0-t+1}$ where t'_0 is the new value of t_0, namely $3l(q') - 2\log p'$. So we conclude that the probability of survival is upper bounded by the mean of the random variable which is $\alpha^{t'_0-t+1}$ if v survives the first phase and 0 otherwise. We refer to this mean as the *bound* and note that it is dependent on what happens outside the neighbourhood of v. ◀

We will come back to the proof of the Theorem, but we first prove the following lemma:

▶ **Lemma 12.** *The bound is maximised when what happens outside the neighbourhood of v is that every neighbour u of v is inhibited from joining the MIS by some external neighbour beeping and no neighbour of v becomes inactive through another vertex (outside $\overline{N}(v)$) joining the MIS.*

Proof. Clearly a vertex ouside $\overline{N}(v)$ joining the MIS can only affect the bound by reducing q which reduces the bound.

Consider any external behaviour E in which some u is not inhibited; we will show that the bound is increased or unchanged if the behaviour is changed to E' in which u is inhibited and there is no change for any other neighbours of v. (In a given graph there may be no such E' but we consider the maximum possible over any graph containing the neighbourhood $\overline{N}(v)$.) We consider fixed beeping decisions of all vertices in $\overline{N}(v)$ except u and show that with these decisions E' gives a value of the bound greater than or equal to that of E. We consider three cases:

- Some neighbour of v which is neither u nor a neighbour of u enters the independent set: Note that this is determined by the fixed beeping decisions and the external behaviour other than as it affects u. Hence this happens for E iff it also happens for E' and in each case the bound is 0.
- Some neighbour of u in $\overline{N}(v)$ beeps:
  p_u will be halved whether or not u is inhibited by E' and so both p' and q' and the probability of survival are the same for E and E'. The bound is identical in the two cases.
- Otherwise:
  Let the value of p' be p_0 if u does not beep and p_1 if u does beep. $p_1 \leq p_0$.
  Let the value of q' be q_0 if u does not beep and is not inhibited, q_1 if it beeps and is inhibited and q_2 if it does not beep and is inhibited. Note that if u beeps and is not inhibited, u enters the independent set and v does not survive. We have $q_1 \geq q_0/4$ since, at most, u's beeping can result in a vertex w halving q_w when otherwise it would have doubled it. Similarly $q_2 \geq q_0/4$ and $q_2 \geq q_0 - 3p_u/2$ since the inhibition results in p_u being halved rather than potentially doubled.
  The bounds are thus $p_u\alpha^{3l(q_1)-2\log(p_1)-t+1} + (1-p_u)\alpha^{3l(q_2)-2\log(p_0)-t+1}$ in the inhibited case and $(1-p_u)\alpha^{3l(q_0)-2\log(p_0)-t+1}$ in the uninhibited case. We claim that the ratio of the inhibited bound to the uninhibited is at least 1. This ratio $\geq \frac{p_u\alpha^{3l(q_1)}+(1-p_u)\alpha^{3l(q_2)}}{(1-p_u)\alpha^{3l(q_0)}}$

(since $p_1 \le p_0$)
Remember that p_u is a power of $1/2$. We consider four subcases:

- $q_0 \le 1/5$: $l(q_1) = l(q_2) = l(q_0) = 0$ and the ratio $\ge (p_u + 1 - p_u)/(1 - p_u) > 1$.
- $1/5 < q_0$ and $p_u \ge 1/8$: We use the bounds $q_1 \ge q_0/4$ and $q_2 \ge q_0/4$ giving that the ratio is at least $(p_u + 1 - p_u)\alpha^{-6}/(1 - p_u) = \alpha^{-6}/(1 - p_u) \ge \alpha^{-6}(8/7) \ge 1$.
- $1/5 < q_0 \le 4/5$ and $p_u \le 1/16$: We use the bounds $q_1 \ge q_0/4$ and $q_2 \ge q_0 - 3p_u/2$ and the fact that for $0 < x \le 15/32$, $(1 - x)^{1/12} > 1 - 4/3(x/12)$ so that the ratio is at least $p_u\alpha^{-6}/(1 - p_u) + (1 - 3p_u/2q_0)^{3\log\alpha} \ge p_u\alpha^{-6} + (1 - 15p_u/2)^{1/12} \ge p_u\alpha^{-6} + (1 - (15p_u/2)/12 \times (4/3)) \ge 1 + p_u(\alpha^{-6} - 5/6) > 1$.
- $q_0 > 4/5$ and $p_u \le 1/16$: Using the same bounds as in the previous subcase the ratio is greater than $\frac{p_u}{1-p_u}\alpha^{-6} + \alpha^{3(l(q_0-3p_u/2)-l(q_0))} > \frac{p_u}{1-p_u}\alpha^{-6} + \alpha^{3(l(4/5-3p_u/2)-l(4/5))}$ and this is the bound already used for the case with $q_0 = 4/5$ and the same value of p_u and so is greater than or equal to 1.

This ends the proof that E' gives a value for the bound at least as great as that for E. The lemma is then proved by a simple induction on the number of uninhibited vertices.

◀

We return to the inductive proof. Using the lemma we will always take $q' = q/2$ giving probability of survival $\le \alpha^{3l(q/2)-2\log p'-t+1}$.

We consider five cases.

- $q \ge 2/5$: We have $l(q/2) = l(q)-1$ and $p' \ge p/2$ giving $P(survival) \le \alpha^{3(l(q)-1)-2(\log p-1)-t+1} = \alpha^{3l(q)-2(\log p)-t}$ as claimed.
- $1/5 \le q < 2/5$ and $p < 1/2$: The probability that a neighbour of v beeps is less than q so that p_v is doubled with probability at least $1 - q$ and halved in the remaining cases. In all cases $l(q/2) = 0$. Hence $P(survival) \le \alpha^{-2\log(p)-t+1}((1 - q)\alpha^{-2} + q\alpha^2)$ and our claim is that it is at most $\alpha^{3\log(5q)-2\log(p)-t}$. That is the claim is valid since $(1 - q)\alpha^{-1} + q\alpha^3 \le \alpha^{3\log(5q)}$ in the range $1/5 \le q < 2/5$. (It is valid at $q = 1/5$ since $4\alpha^{-1} + \alpha^3 < 5$ and at $q = 2/5$ since $3\alpha^{-1} + 2\alpha^3 < 5\alpha^3$; between these two limits, the left hand side is linear and the right hand side $((5q)^{3\log\alpha})$ has a negative second derivative so the inequality holds there also.)
- $1/5 \le q < 2/5$ and $p = 1/2$: With probability greater than $1-q$ no neighbour of v beeps and then v has probability $1/2$ of entering the independent set; otherwise p_v remains $1/2$. On the other hand, if a neighbour does beep, p_v becomes $1/4$. In all cases $l(q/2) = 0$. Thus the probability of survival $\le \alpha^{2-t+1}((1 - q)/2 + q\alpha^2)$ and the claim is that it is at most $\alpha^{3log_2(5q)+2-t}$. That is the claim is valid if $(1 - q)\alpha/2 + q\alpha^3 \le \alpha^{3\log(5q)}$ a weaker condition than in the previous case.
- $q < 1/5$ and $p < 1/2$: The probability that a neighbour of v beeps is less than $1/5$ so that p_v is doubled with probability at least $4/5$ and halved in the remaining cases. In all cases $l(q)$ decreases or is unchanged. Hence $P(survival) \le \alpha^{3l(q)-2\log(p)-t+1}((4/5)\alpha^{-2} + (1/5)\alpha^2)$ and this is less than $\alpha^{3l(q)-2\log p-t}$ as claimed, again since $4\alpha^{-1} + \alpha^3 < 5$.
- $q < 1/5$ and $p = 1/2$: With probability greater than $4/5$ no neighbour of v beeps and then v has probability $1/2$ of entering the independent set; otherwise p_v remains $1/2$. On the other hand, if a neighbour does beep, q decreases and p_v becomes $1/4$. Hence $P(survival) \le (2\alpha^{3l(q/2)-2\log(1/2)-t+1} + \alpha^{3l(q/2)-2\log(1/4)-t+1})/5 \le \alpha^{3l(q)-2\log(1/2)-t+1}(2 + \alpha^2)/5$ which is at most $\alpha^{3l(q)-2\log(1/2)-t}$ as claimed since $2 + \alpha^2 < 5\alpha^{-1}$.

This completes the proof of the theorem.
We end this section by the following corollary:

▶ **Corollary 13.** *The number of phases taken by Jeavons et al.'s algorithm on any graph with n nodes is, in fact, less than $76 \log n$ w.h.p.*

Proof. Since initially $p_v = 1/2$ and $q_v < n/2$ where the graph has n vertices, we conclude that $t_0 < 3\log(5n/2) - 2\log(1/2) < 3\log n + 6$ so that after $t = 76 \log_2 n$ phases, every vertex v has probability n^{-2} of still being active and therefore the algorithm has terminated with probability $1 - o(n^{-1})$. ◀

▶ Remark. The number of phases before the probability is $1 - o(n^{-1})$ compares well with the value of $K_0 \geq e^{25}$ proved in [8].

A.5 The Case of the 2-hop MIS Las Vegas Algorithm in $B_{cd}L_{cd}$

The 2-hop MIS algorithm simulates the MIS algorithm in the square of G; knowing that the complexity depends only on the number of vertices of the graph we deduce from the previous section:

▶ **Theorem 14.** *The number of phases taken by the 2-hop MIS algorithm on any graph with n nodes is less than $76 \log n$ w.h.p.*

B Missing proofs

This appendix section provides the missing proofs of Section 5.

Proof of Lemma 4. By induction on t. Clearly the claim is true for $t = 0$. We suppose that E_{t-1} is true and we consider probabilities conditional on the values of B_{t-1} for $is - 1$, is, $is + 1$, $(i + 2m)s - 1$, $(i + 2m)s$ and $(i + 2m)s + 1$.

We will show that the probability that $b_t^{is} = b_t^{(i+2m)s}$ and $b_t^{is-1} = b_t^{(i+2m)s-1}$ and $b_t^{is+1} = b_t^{(i+2m)s+1}$ is at least 2^{-3}.

The three events:

- $b_t^{is} = s_b^{(i+2m)s}$
- $b_t^{is-1} = b_t^{(i+2m)s-1}$
- $b_t^{is+1} = b_t^{(i+2m)s+1}$

are independent.

For the first, u_{im} and $u_{(i+2m)s}$ started in the same state and have sent and heard identical signals. Thus they have the same probability of beeping at the next round and so have probability at least $1/2$ of either both beeping or neither.

For the second, the two chains $(u_{(i-1)s} \cdots u_{is-1})$ and $(u_{(i+2m-1)s} \cdots u_{(i+2m)s-1})$ started in the same states, have received the same signals from u_{is} and $u_{(i+2m)s}$, and have sent the same signals. Thus, again the two vertices u_{is-1} and $u_{(i+2m)s-1}$ have the same conditional probability of beeping and so probability at least $1/2$ of making the same choice.

The argument for the third event is identical.

This proves that the three events happen with probability at least 2^{-3} yielding that the probability of event E_t is lower bounded by 2^{-3t}. ◀

Proof of Theorem 5. For simplicity we consider wheels (m, s) where s is a power of 2 and $m = 2^{4s-2}/s$ so that $s = \log_2 n/4$. We consider a computation on this wheel without specifying whether it is the odd or even wheel. By Lemma 4, the probability that a given spoke i breaks symmetry within time $s - 1$ is at most $1 - 2^{3-3s} < exp(-2^{3-3s})$ and this is independent for all spokes so that the probability that every spoke breaks symmetry in the even case in time $s - 1$, is at most $exp(-2^{3-3s}m) = exp(-2^{s+1}/s) < 1/4$. Hence the probability that the algorithm halts and some spoke fails to break symmetry is greater than $1/2$. If, in the even case, spoke i fails to break symmetry, vertex u_i hears the same signals from its neighbours in the odd and even cases and, so, if it terminates the algorithm in this time, it has the same probability of deciding collision in the two cases. Hence it gives the wrong response in one case with probability at least $1/2$. Hence there is a vertex which gives the wrong response in the odd or even case with probability greater than $1/4$.

Thus if an algorithm halts in time $o(\log n)$ with probability $\geq 3/4$, for sufficiently large n it halts in time less than s and so its probability of giving an incorrect result is at least $1/4$ for some initial conditions. It follows that the same is true for any algorithm halting in expected time $o(\log n)$. ◀