ONLINE, COMPUTABLE, AND PUNCTUAL STRUCTURE THEORY

MATTHEW ASKES AND ROD DOWNEY

Abstract. Several papers (for example [7, 23, 42]) have recently sought to give general frameworks for online structures and algorithms ([4]), and seeking to connect, if only by analogy, online and computable structure theory. These initiatives build on earlier work on online colouring and other combinatorial algorithms by Bean [10], Kierstead, Trotter et. al. [48, 57, 54] and others, as we discuss below. In this paper we will look at such frameworks, and illustrate them with examples from the first author's MSc Thesis ([58]). In this thesis, Askes looks at online, adversarial online, and strongly online algorithms and structures. Additionally, we prove some new theorems about online algorithms on graphs of bounded pathwidth as well as some new results on punctually 1-decidable structures.

Contents

1. Introduction

1.1. **Offline and online.** In most of classical mathematics, we tend to think of objects being presented as fully formed wholes. This is particularly true in combinatorics where we consider a graph (or a partial ordering, etc.) as *offline*. That

Both authors were partially supported by Marsden Fund of New Zealand. This paper was written for the *John Crossley Logic Colloquium, 2022.* Crossley was Downey PhD adviser back in the late 1970's, and Downey has many fond memories of his time at Monash with the logic group John established there.

is, given as a set of vertices with edges between them. In computability theory, we think of objects as being formed with time, and the key ideas are dealing with approximations to the truth. Proofs tend to be a kind a game played between the us and the hostile universe, where the goal tends to be the discovery of a strategy. Anyone who has worked in classical complexity theory knows that, for example, polynomial time reductions are usually quite static. Indeed they are defined in terms of the given input structure. In computability theory, constructions tend to be wildly adaptive and highly recursive.

However one uniting area is that of *online algorithms*. The online version of the famous NP-complete BIN PACKING problem is the minimization of the number of bins needed to pack objects being given as a stream $o_1, o_2, \ldots$, and we put the first n into bins before we are given o_{n+1}. (See Karp [44]). You are in an *online* situation and this is the ONLINE BIN PACKING problem. The "first fit" method is well-known to give a 2-approximation algorithm for this problem. That is, the *performance ratio* is ≤ 2, where this is the fraction of the online cost (the number of bins used for n objects) divided by the number of bins for an optimal solution

$$\frac{C_{\text{online}}(n)}{C_{\text{offline}}(n)}$$

We remark that this ratio is a standard measure of the efficacy of any online minimization algorithm (Sleator and Tarjan [66]). Alternatively imagine you are a scheduler, and your goal is to schedule requests within a computer for memory allocation amongst users. Again you are in an online situation, but here you might want to change the order of allocation depending on priorities of the requests. Or from algorithmic randomness, you have a (computable) KC-set of requests of the form $(2^{-n_i}, \sigma_i)$ with $\sum_{i=1}^{\infty} 2^{-n_i} \leq 1$, and need to build a prefix-free Turing machine M with strings τ_i such that $|\tau_i| = n_i$ and $M(\tau_i) = \sigma_i$. Then the proof from e.g. Downey and Hirschfeldt [27] Theorem 3.6.1, is online in the sense that for each request at step i we generate the string τ_i. Another example from algorithmic randomness is a computable martingale m[1] betting on bits of $\alpha \in 2^\omega$, where it succeeds if $\limsup_n m(\alpha \upharpoonright n) = \infty$, and α is computably random if no computable martingale succeeds. An online algorithm is one which acts on a input which is given piece by piece in a serial fashion and with an unknown future. In the case where the input is finite, Karp [45] suggested this as a sequence of "requests" $r_1, r_2, \ldots$ with the algorithm f specifying an action $f(r_1), f(r_1 r_2), \ldots$. In the corresponding *offline* version (at least in the finite case), the whole input is known in advance.

As per Albers [4], we know that that there are many examples of such online algorithms in computing: examples include insertion sort, perceptron, paging, job shop scheduling, ski rental, navigation with only local understanding, etc.

On the other hand, there seems no general *theory* which we can use as a conceptual basis for the theory of (finite and infinite) online algorithms, and online structures. Books on online algorithms tend to be taxonomies. In the next subsections we will look at various recent general settings for such studies [2].

[1]Which the reader might recall is a computable function $m : 2^{<\omega} \to \mathbb{R}^+ \cup \{0\}$ such that $m(\sigma) = \frac{m(\sigma 0) + m(\sigma 1)}{2}$.

[2]The reader might wonder why we would need general models. Online algorithms are everywhere without such a model. We see the situation as being akin to the development of, for example, asymptotic counting of computation steps as a measure of computational complexity by Hartmanis and Stearns in 1965 [40] (amongst others). Before this, people ran algorithms and had

1.2. **Turing computable mathematics.** In this area, we study structures with computable (or perhaps sub-computable) presentations. To wit, $\mathcal{A} = \langle A, R_i, f_j, c_k \mid i \in D, j \in B, k \in C\rangle$ where the R_i ($i \in D$) are a uniformly computable set of relations, f_j a uniformly computable set of functions and c_k a uniformly computable set of constants. For instance, a computable field $\langle F, +, \cdot, 0, 1\rangle$ has F a computable set (typically identified with $\mathbb{N}$), $+$ and $\cdot$ computable functions so that, for example, $x \cdot z = y$ would be a computable relation on the set of triple $\langle x, y, z\rangle$. The idea of an algorithm is implicit in virtually all pre-19th century mathematics. It was in the early 20th century that authors such as Borel, Dedekind, Herrmann and others realized intuitively that processes might not be algorithmic. For example, we have the following quote from one of the first papers in what might be called modern computable mathematics. Frölich and Shepderdson [32] studied algorithmic procedures in field theory, such as whether there is an algorithm to classify the algebraic closure. This paper clearly shows the historical context of the subject, the clear intuition of van der Waerden (which apparently came from Emmy Noether's lecture notes) and the fact that isomorphic computable structures (here fields) can have distinct algorithmic properties, and hence cannot be computably isomorphic. Here we quote from the abstract.

> "Van der Waerden (1930a, pp. 128–131) has discussed the problem of carrying out certain field theoretical procedures effectively, i.e. in a finite number of steps. He defined an 'explicitly given' field as one whose elements are uniquely represented by distinguishable symbols with which one can perform the operations of addition, multiplication, subtraction and division in a finite number of steps. He pointed out that if a field K is explicitly given then any finite extension K' of K can be explicitly given, and that if there is a splitting algorithm for K i.e. an effective procedure for splitting polynomials with coefficients in K into their irreducible factors in $K[x]$, then (1) there is a splitting algorithm for K'. He observed in (1930b), however, that there was no general splitting algorithm applicable to all explicitly given fields K, [...] We sharpen van der Waerden's result on the non-existence of a general splitting algorithm by constructing (§7) a particular explicitly given field which has no splitting algorithm. We show (§7) that the result on the existence of a splitting algorithm for a finite extension field does not hold for inseparable extensions, i.e. we construct a particular explicitly given field K and an explicitly given inseparable algebraic extension $K(x)$ such that K has a splitting algorithm but $K(x)$ has not."

In modern terms Frölich and Shepderdson [32] showed that the halting problem is many-one reducible to the problem of having a splitting algorithm.

We remark that Turing's first paper [69] was not concerned with countable structures as above but with computational processes on the reals (at least the field of computable reals). Suppose that we are wishing to investigate computation on a smooth space of analysis, such as the real numbers between 0 and 1. There is no essential problem extending the definition of a computable structure to a continuous one with a *computable dense set*. Following the modern "type 2" definition, we think of elements of, for instance, a computable metric space as being *represented* by fast Cauchy sequences of rationals and computable functions as being those which are uniformly computable operators $\Phi^\alpha(n) = \beta(n)$, meaning that Φ represents the computation of f with $\Phi^\alpha(n)$ computing $f(\alpha) = \beta$ to precision 2^{-n}.

an intuitive understanding of complexity, but the development of a framework allows for formal analysis. Similar comments apply to, for example, the development of parameterized complexity by the second author and Fellows [25, 24]. People already had been pre-processing in algorithmics, but the explicit identification of a complexity theory based on the multivariable contribution of the various aspects of the input (in place of simply the overall size) hastened the introduction of standard tools for such algorithmics ([25, 63]). We have similar hopes for online algorithmics.

Notice that it is immediate that a function f being computable implies that it is continuous. The converse is somewhat true, in that f is continuous means that there is some X such that f is computable relative to X. That is because we only need the information as to where balls or radius 2^{-m} around rational points are sent to, the countable information X given by balls around dense sets specifies f.

A nice arena of computable mathematics is computable combinatorics. We know that all planar graphs are 4-colourable. What about computable planar graphs? The following is more or less folklore and certainly implicit in Bean [10]. The reference given is where it was—more or less—first explicitly stated (we will clarify this comment later.).

Theorem 1.1 (Gyárfás and Lehel [38])**.** *There is a computable forest which cannot be finitely coloured by a computable algorithm.*

Proof. (sketch) We break the forest into components C_e where we defeat the e-th partial computable colouring φ_e. The k-th task for C_e is to make φ_e use at least k colours, or, equivalently use the k-th colour. In this component, we first introduce a vertex v_1, wait for φ_e to colour it, and then introduce another v_2, waiting for φ_e to colour it also. Assuming he has played with only one colour, then these are both the same, and we can add one vertex v_3 joined to one of them v_2. Again after φ_e colours it he must choose at worst colour 2. Then if we join v_4 to v_1 and v_3, the opponent must choose colour 3 for v_4. Thus we have a way of making sub-components which use at least 3 colours. Using enough of these components, we can join up as trees to force him to use 4 colours etc. $\square$

The reader might note that if we are given a computable tree (with the knowledge it is a tree T) then it must be computably 2-colourable. This is because we can see where each new vertex lies on the tree by waiting for the tree to become connected. That is, at stage s we will have T_s and a new vertex x is added, we can run the enumeration of $T = \bigcup_s T_s$ till there is a path from x to T_s.

Bean gave a similar argument to that of Theorem 1.1 to demonstrate the following.

Theorem 1.2 (Bean [10])**.** *There is a computable 3-colourable connected planar graph which cannot be coloured computably with a finite number of colours.*

To prove this, Bean had layers of planarity all joined to a single vertex, and growing outwards. The component C_e to defeat φ_e forced one new colour per layer. So the structural parameters of planarity and being a tree don't seem to help with computable colouring. However, sometimes structural parameters do help.

Definition 1.3.
(1) A graph G is an *interval graph* if there is a mapping $\iota : V(G) \to \{(a,b) : a < b \wedge a, b \in [0,1]\}$ such that $xy \in E(G)$ iff $\iota(x) \cap \iota(y) \neq \emptyset$.
(2) Let c be the cutwidth -1 of this interval decomposition, that is, the maximum intersection of intervals (a,b) in the range of ι. Then the quantity $c-1$ is called the *width* if the decomposition, and the width of G is the minimum width of the widths of all decompositions. It is not hard to see that this width is the same as the size of the maximum clique minus 1.

The "-1" is to make sure that paths have width 1. But there are graphs of width 1 which are not paths, such as "fuzzy balls", that is a vertex v at the centre of disjoint edges all incident with v. Indeed a connected G has pathwidth 1 iff G is path consisting of such fuzzy balls (this is called a *caterpillar graph*).

If G is a subgraph of H with the same vertex set, and H is an interval graph, then we say that H is an *interval completion* of G. The interval width of G can be taken as the minimum interval width of an interval completion of G. Interval width and *pathwidth* are the same concepts, where G has a path decomposition of width k iff there is path $P = \{p_1, \ldots, p_d\}$ so that $p_i p_{i+1}$ is an edge, and a function h associating a set of vertices of G with each element of P, such that

(1) Every $v \in V(G)$ is in some $h(p_i)$,
(2) If $xy \in E(G)$ then there is some i with $x, y \in h(p_i)$, and
(3) If $x \in p_i$ and $x \in p_j$ for $i < j$, then $x \in p_q$ for all $i \leq q \leq j$. (Interpolation property.)

The width is then the maximum size of $|h(p_i)| - 1$ for $1 \leq i \leq n$, and again the pathwidth of G is the minimum over all path decompositions. Each set $h(p_i)$ is usually referred to as a *bag*, $B_{p_i} = B_i$. Note that, if G has pathwidth k then if we take any path decomposition with bags $B_1, \ldots, B_d$, then form the graph G^* by making each of these bags a clique (specifically a $k+1$-clique if the width is k), then G^* also has pathwidth k and is called a k-path, so that paths of pathwidth k are *partial k-paths.*

It is routine to show that G has a path decomposition of with k iff it has an interval decomposition of width k. Following standard practice, we will let I_x denote $\iota(x)$ in an interval decomposition, and hence we will use these terms interchangeably. If a graph has pathwidth k then it can be coloured with $k+1$ many colours simply by taking the path decomposition and colouring vertices using the greedy algorithm.

Theorem 1.4. [3] *If G is a computable graph of pathwidth k, then G can be computably coloured by $3k+1$ many colours.*

Proof. This is proven by induction on the width k, and for each k we recursively construct an algorithm A_k. If $k = 1$ then G is a caterpillar graph and we can use greedy minimization which will use at most 3 colours. So suppose $k > 1$, and let G_n have vertices $\{v_1, \ldots, v_n\}$. The computable algorithm A_k will have computed a computable partition of G, which we denote by $\{D_y \mid y < k\}$. We refer to the D_y as *layers.* Consider v_{n+1}. If the pathwidth of $G_{n+1} = G_n \cup \{v_{n+1}\}$ is $< k$, colour v_{n+1} by A_{k-1}, and put into one of the cells D_y, for $y < k-1$ recursively. (In the case of pathwidth 1, this will all go into D_0.) We will be colouring using using the set of colours $\{1, \ldots, 3k-2\}$.

If the pathwidth of G_{n+1} is k, consider H_{n+1}, the induced subgraph of G_{n+1} generated by $G_{n+1} \setminus D_k$. If the pathwidth of H_{n+1} is $< k$, then again colour v_{n+1} by A_{k-1}, and put into one of the cells D_y, for $y < k-1$, recursively, and colour using the set of colours $\{1, \ldots, 3k-2\}$. If the pathwidth of H_{n+1} is k, then we put v_{n+1} into D_{k-1}. In this case, that is in D_{k-1}, we will use first fit using colours $3k - 2 < j \leq 3k+1$.

The validity of this method follows from the fact that the maximum degree of vertices restricted to D_{k-1} is 2, and induction on k. Assume that A_{k-1} is correct and colours the subgraph of G_n induced by the layers $\{D_y \mid y < k-1\}$ using colours $\{1, \ldots, 3k-2\}$.

[3]After Kierstead and Trotter [54] and Downey and Fellows [25]. Strictly speaking, this result was proven by Kierstead and Trotter *only* for interval graphs, and hence for k-paths. Downey and Fellows state the result (incorrectly as $3k-2$ rather than $3k+1$) for pathwidth, but it is not altogether clear that the proof is correct. We show that we can modify one of Kierstead-Trotter proofs so that it works for partial k-paths, i.e. pathwidth k graphs.

Note that the construction ensures that the pathwidth of this subgraph H_k is at most $k-1$. Moreover, induction on n ensures that A_{k-1} would colour the vertices of the subgraph of G_n induced by $\{D_y \mid y < k-1\}$ using colours $\{1, \ldots, 3k-2\}$ assuming the vertices of G_n in D_{k-1} did not exist, the same as A_k colours them. To see this, assuming that it is true up to step n, then if v_{n+1} is added to D_{k-1} then there is nothing to prove, and if it is added to $\cup_{j<k-1} D_j$, then this step will invoke A_{k-1} and since the colours of D_{k-1} will exceed $3k-2$, they have no effect on the colouring of the subgraph of G_{n+1} induced by $\cup_{j<k-1} D_j \cup \{v_{n+1}\}$.

Suppose that that v_{n+1}'s addition to G_n has pathwidth k. Now consider a path decomposition $B_1, \ldots, B_q$ of G_{n+1}. Suppose, for a contradiction, that the degree of $v = v_{n+1}$ in D_k is ≥ 3. Thus there are x, y, and z in D_k which are each connected to v. Without loss of generality, let's suppose that that they were added at stages $s_x < s_y < s_z \leq n$. Since each is in D_k, when we added them to D_k, we could not have added them to D_y for $y < k-1$. Since they were not added to such D_y it follows that at the stages they were added, they made the pathwidth of the relevant H_s ($s \in \{s_x, s_y, s_z\}$ to be k. Consider s_x. As the pathwidth of H_{s_x} was k, there must be some bag in any path decomposition of G_{s_x}, consisting of only members of G_{s_x} which has size $k+1$, and containing x. For $t > s_x$, this must still hold, that is, x must be in, at stage t, a bag of size $k+1$ consisting only of elements of G_{s_x}. For suppose this was not true at stage t. The pathwidth of G_t is k, and has bags $P_1, \ldots, P_v$, say. Now delete all of the elements of $G_t \setminus G_{s_x}$ from the bags forming bags $P'_1, \ldots, P'_v$. This is a path decomposition of G_{s_x}, and hence must have pathwidth k, so there must be one of size $k+1$ containing x, and it only consists of elements of G_{s_x}.

Consider s_y. Since the pathwidth of H_{s_y} is k, it follows that s_y must be in a bag of size k in the path decomposition of H_{s_y} containing none of D_{k-1}. In particular, in any path decomposition of G_{s_y}, x and y must appear in bags Q_x and Q_y, respectively, of size k with $x \notin Q_y$ and $y \notin Q_x$,

Using the same reasoning, as above, this fact must hold also for each stage $t > s_y$. So we can conclude, using the same reasoning, that at stage $n+1$, x, y, z, and v are all in bags of size k, B_x, B_y, B_z, B_v, where $x \notin B_y \cup B_z \cup B_v$, and similarly for y, z and v.

Now consider B_v. Since xv is an edge, x and v lie together in some bag B_{xv}. If B_{xv} is left of B_v but B_{xv} is right of B_v we get a contradiction, since this would put x into B_v, by the interpolation property of pathwidth. So B_{xv} and B_x both lie, without loss of generality left of B_v. Similarly B_{yv} and B_y must lie on the same side, and this must be right. For if there were both left of B_v, then the interpolation property would make either B_x or B_y contain y of x respectively (considering the relevant orientations of B_x and B_y). But now we get a contradiction, since B_z cannot be either right or left of B_v without one of the B_x, B_y, or B_z containing a forbidden element. Thus, within D_{k_1} the degree of v is at most 2. $\square$

We remark that the proof of the theorem above gives an algorithm which is linear time (as k-Pathwidth is linear time FPT), but is inefficient as the constants for the pathwidth algorithm (Bodlaender's Algorithm for treewidth.) are of the order of 2^{35k^3} which is pretty horrible. This algorithm is invoked at each step. We don't know the best complexity for the following (online) promise problem, which is highlighted by such considerations.

Input: An online graph G, and a vertex v and a graph H with vertices $V(G) \cup \{v\}$ G a subgraph of H.
Promise: G has pathwidth k.
Parameter: An integer k.
Question: Does H have pathwidth k?

We remark that Kierstead and Trotter [54] proved that Theorem 1.4 is sharp, in that for all k, there is a computable graph G of pathwidth k that cannot be $3k$ coloured. For example, for pathwidth 1, we can force 3 colours, begin with an edge uv causing colours 1 and 2, and a disjoint one u_1, v_1 causing 1 and 2. Add a new vertex x joined to make a path with the vertices of each colour. Then x needs a third colour. Now again take two disjoint edges using 2 colours again and add one further vertex y joined to each of the vertices of the three colours. This still has pathwidth 1 and needs a fourth colour. The general case follows by similar ideas, and induction.

1.3. **Highly computable graphs.** The reader may note that the arguments we have used to force the number of colours the computable colourer must use, mostly rely on the fact that we don't know the answer to Σ_1^0 questions about the structure. In particular, if we have a vertex $v \in G_s$ we know the neighbours of v in G_s, but not necessarily in G, as later there might appear $y \in G_t$ for $t > s$ with $yv \in E(G)$. For the remainder of this section we assume that graphs are locally finite.

Definition 1.5. We say that G is *highly computable* (historically "highly recursive") if there is a computable function f such that, for all v, $f(v)$ computes the degree of v in G.

There have been a number of results proven about computable combinatorics of highly computable graphs. We will see some later in § 6 but as an easy example, we give the following well-known result.

Theorem 1.6 (Bean [10])**.** *If G is a highly computable graph with chromatic number $\chi(G)$ then G can be computably coloured with $2\chi(G)$ many colours.*

Proof. Let $G = \lim_s G_s$ be a k-colourable highly computable graph.

We define the colouring at odd and even stages. In the even stages we use colours $1, \ldots, k$ and in the odd stages $k+1, \ldots, 2k$.

We colour $G_0 = \emptyset$ with no colours.

For the even stage $2s+2$: Suppose we have coloured G_{2s+1} using $2k$ colours. Let v be the next vertex presented. Since we can compute the degrees of vertices in G, we can compute $N[G_{2s+1} \cup \{v\}]$, the whole induced subgraph of $G_{2s+1} \cup \{v\}$ together with any edges joined to it. Let $G_{2s+2} = N[G_{2s+1} \cup \{v\}]$. At stage $2s+2$ we can see all of G_{2s+2}. Let $\bar{G}_{2s+2} = G_{2s+2} \setminus G_s$. We assign every vertex in $\bar{G}_{2s+2}$ a colour from $1, \ldots, k$. This can be done using brute force as G is k-colourable and $\bar{G}_{2s+2}$ is finite.

The odd stage is analogous, but we colour $\bar{G}_{2s+1}$ with colours from $k+1, \ldots, 2k$
This gives a colouring of G as $\bar{G}_s$ is disconnected from $\bar{G}_{s+2}$. □

We also have the following tight extension of Bean's Theorem by Schmerl.

Theorem 1.7 (Schmerl [64])**.** *If G is a highly computable, k-colourable graph, then G is computably $(2k-1)$-colourable.*

We will look at this again in § 6.

2. Online structures I

One of the first attempts to give a general framework for online structures, focused upon the intuition that online decisions in practice have *lack of delay.* That is, we need to pack the object into some bin immediately, before the next one is presented to us (as in the Bin Packing example). This led to a theory of online structures and algorithms we generally referred to as *punctual* structure theory.

What is the most general reasonable form of "punctual"? In [7] Bazhenov et. al. gave several pages of analysis as to why we chose to interpret punctuality as *primitive recursive.* That is, we chose primitive recursive as a *unifying abstraction* of the notion of lack of delay.

There are several related reasons for this choice. There had been some investigations into *subcomputable* structure theory. Khoussainov and Nerode [46] initiated a systematic study into automatically presentable algebraic structures; but these seem quite rare. In their approach we considered functions as relations, and for example, "$x + y = z$" in an abelian group would be modeled by a three tape automaton which accepted with x, y, z on the relevant tapes. The additive group of the rationals is not automatic [68]. Approaching lack of delay using finite automata is highly sensitive to how we define what we mean by automatic. For example treating a function as a relation yields quite a different kind of automatic presentation. See [31] for an alternate approach to automatic groups. Cenzer and Remmel, Grigorieff, Alaev, and others [16, 37, 3, 1] studied polynomial time presentable structures. We omit the formal definitions, but we note that they are sensitive to how exactly we code the domain.

In the end, a nice unifying abstraction was introduced by Kalimullin, Melnikov and Ng [43, 42], which they called *fully primitive recursive*, but has subsequently been re-named as *punctual.*

Definition 2.1 (Kalimullin, Melnikov and Ng [42])**.** A countable structure is *fully primitive recursive* or *punctual* if its domain is $\mathbb{N}$ and the operations and predicates of the structure are (uniformly) primitive recursive. (For simplicity, we assume the languages are finite.)

We stress that the domain of the structure is $\mathbb{N}$, which codes the timestamps of the enumeration of the structure. Various pathologies occur if arbitrary primitive recursive subsets are allowed as domains as mentioned in [7]. There are several arguments supporting the choice of primitive recursive as a unifying abstraction. First, we have a nice version of the Church-Turing Thesis for these functions as being those computable without while loops. They have nice closure properties, and are relatively insensitive to coding. In [7], Bazhenov et. al. also noted that many results stated in terms of primitive recursion, can likely be pushed to polynomial time structures. Furthermore, some of our counterexamples can in fact be stated in terms of *any* class with sufficiently nice closure properties; e.g. for a class of total computable functions having a uniformly computable enumeration and closed under composition and primitive recursion. However, this does not mean that our choice of primitive recursive algorithms as a central model is arbitrary.

For example, from a logician's point of view these structures are completely natural: One of the *fundamental results* of computable structure theory is that:

A decidable theory has a decidable model.[4]

The proof of this elementary fact is to observe that the Henkin construction is effective, in that if the theory is decidable then the constructed model is decidable as a model. Many standard computable structures come from decidable theories.

Most natural decidable theories are *elementary* decidable in that the decision procedures are relatively low level. We have to go out of our way to have natural decidable theories whose decision procedures are not primitive recursive. In [7], Bazhenov et. al. observed that a theory with a primitive recursive decision procedure has a model which is decidable in a primitive recursive sense; meaning that it has primitive recursive Skolem functions providing witnesses quickly (more on this below when we look at primitive recursive 1-decidability and honesty).

We remark that it is easy to give intuition why punctual structure theory is not a simple generalization of Turing computable structure theory. Consider colouring a computable line. We are given this as a collection of vertices. We are given v_0 and colour it 1. We are given v_2 and should we colour it 1 or 2 if v_0v_1 is not an edge. In the computable case, we can wait for v_0 and v_1 to be connected, since we are dealing with a path and only then decide. In the punctual case, even with primitive recursive delay of $t(2)$ many steps, where t is the relevant punctual time bound, it may be that v_0 and v_1 remain disconnected. Then whatever we choose for v_2 this colour might then force up the number of colours from 2 to 3.

Bazhenov et. al. [7] showed that the theory of punctual structures is surprisingly rich. There is a fascinating interplay between computable and punctual structure theory. For example, consider the following Theorem.

Theorem 2.2 (Kalimullin, Melnikov, and Ng [42]). *The following structures all have computable presentations iff they have fully primitive recursive ones.*

(1) *Linear orderings (Grigorieff* [37]*)*
(2) *Boolean algebras*
(3) *Equivalence structures*
(4) *Torsion-free abelian groups*
(5) *Abelian p-groups*
(6) *Locally finite graphs*

Proof. We sketch a couple of proofs as they will be useful in what follows. Let $\langle E, \equiv_E\rangle$ be a computable equivalence relation. We build a primitive computable $\langle F, \equiv_F\rangle \cong \langle E \equiv_E\rangle$. One way to do this is to break it into several cases. Suppose that there is an infinite equivalence class C_1 in E. Then to build F, we will build a primitive computable equivalence class D_1 to which C_1 will be mapped. That is, at each stage, whilst we are waiting for new elements to be enumerated into E, along with their equivalence classes, we can always put the next element into $D_1[s+1]$ and always be safe. Otherwise when a new element x arrives in E_s we will wait for a stage $t > s$ when we know exactly which of the stage s equivalence classes $C_1, \ldots, C_{p(s)}$, of E_s, x lies. If it lies in one of those D_j with a current image in F_t, then add a new element to increase the cardinality of $D_j[t+1]$. If not, then begin a new equivalence class in D_{t+1} corresponding to one which contains x in E.

[4]Recall that a complete, first-order theory in a computable language is decidable if the collection of all Gödel codes of its sentences forms a computable set. A model upon the domain $\mathbb{N}$ is decidable if there is a Turing computable algorithm which, given a first-order formula with parameters from the domain of the structure, can decide whether the formula holds in the structure.

This “marking time and waiting to see what happens” is quite useful. It can also be adapted to the situation where all equivalence classes of E are finite, because at every stage we can begin a new equivalence class to mark time, and later include it in the range of the intended isomorphism to F.

A similar strategy can be used for linear orderings (This is not Grigorieff’s strategy.) Given a computable L we need to build primitive recursive $\hat{L} \cong L$. Then we can break into cases as follows. By Rosenstein [62], L has a computable subordering or order type ω^*, ω, $\omega + \omega^*$ or $\omega + \eta \cdot \zeta + \omega^*$, where η is the order type of the rationals and ζ that of the integers. So suppose, for instance, L has a computable subordering L_1 of type ω. Then we can use this to mark time, building a subordering of $\hat{L}$ of this type, and then paying attention when other element are slotted into L. The other cases are similar. □

All such proofs seem to rely on enumerating a “nice” subset of the structure.

Remark 2.3. The reader should note that the in the proofs above, not only do we construct a punctual copy, but also the isomorphism from the computable copy to the punctual one is a computable isomorphism.

Whilst many structures have punctual copies of computable models, some like graphs and partial orders can have computable copies such that no punctual structure is isomorphic to any of the copies. For instance, we have the following:

Theorem 2.4 (Kalimullin, Melnikov, and Ng [42])**.** *In each of the classes below, there are examples of computably presentable structures without punctual presentations.*

(1) *Torsion abelian groups*
(2) *Undirected graphs*
(3) *Archimedean ordered abelian groups.*

The proof of Theorem 2.4 has no common technique for the parts.

Proof. We offer a proof for (2). We can enumerate all primitive recursive graphs M_e for $e \in \omega$. The graph G we build will consist of disjoint loops L_i. To diagonalize M_0, and define our first loop, we have 0 in M_0 and we run M_0’s enumeration till we find it giving us the fact that 0 is in a n_0 loop for suitably chosen n_0. If M_0 has such a loop then we make L_0 have size bigger than n_0, and then promise that all loops in G will have size bigger than n_0, diagonalizing M_0. If M_0 does not enumerate such a loop, then we can make a loop of size smaller than n_0 to put in G. At the next stage, we can attack again using a bigger $n_0[s + 1]$ and if ever the opponent gives us a loop, we can diagonalize since all of our loops will have size smaller than $n_0[t]$ but if he returns, then all subsequent loops will be bigger. For more than one requirement the action is similar depending on the guess as to the outcome, and simply play a strategy “behind the $n_0[s]$ boundary, when it assumes that it is tending to infinity.” □

You might ask if there is some reasonable description of the structures with punctual presentations. If $\{A_e \mid e \in \omega\}$ lists partial computable algebraic structures, then we have the following saying that those with punctual presentations are unclassifiable.

Theorem 2.5 (Bazhenov et. al. [8])**.** *The index set* $\{e : A_e$ *has a punctual presentation*$\}$ *is* Σ^1_1*-complete.*

The theorem actually works for "automatic" or "polynomial time" in place of "punctual", and involves direct coding of Turing machines in a novel way. There is a very rich structure of the punctual *degrees* of punctual copies of a computable isomorphism type, once we define the appropriate reducibility. One notable result is that graphs are *not* universal structures for punctual structures, but there is are universal structures built from one function symbol:

Theorem 2.6 (Downey et. al. [26])**.** *The class of structures with only one binary functional symbol is punctually universal*

Finally there are strange examples of structures which are, for example, punctually categorical but not computably categorical. We refer the reader to [7] for more on this topic.

Perhaps inspired by the notion of being highly computable, people have studied punctual structures with stronger online characteristics. To have a notion of punctually decidable we need punctual Skolem functions, as knowing an existential formula being true is not much help unless you have access to witnesses.

In particular a formula $\exists \overline{x}\, \Phi(\overline{a}, \overline{x})$ is punctually decidable in a structure $\mathcal{A}$, if there exists a primitive recursive g, and if $\overline{a} \in \mathcal{A}$, then if $\exists \overline{x}\, \Phi(\overline{a}, \overline{x})$ is true in $\mathcal{A}$, $g(\overline{a}) = \overline{b} \in \mathcal{A}$ and $\Phi(\overline{a}, \overline{b})$, and if $\neg\exists \overline{x}\, \Phi(\overline{a}, \overline{x})$ in $\mathcal{A}$ then $g(\overline{a}) = \bot$.

Then, following [9], we can say that $\mathcal{A}$ is *punctually 1-decidable* if it is punctually decidable for all Σ^0_1-formulae (and hence all Π^0_1 formulae). Little is known about this class. We mention a couple of results. We have seen above that every linear ordering with a computable is computably isomorphic to one with a punctual copy. This fails for 1-decidability. Note that a punctual linear ordering is punctually 1-decidable iff it has punctual successivities (e.g. Blinov [11]), an analog of the fact that L is 1-decidable iff the adjacencies of L are computable.

Theorem 2.7. *There is a computable 1-decidable linear ordering of order type $\mathbb{Z}^\omega$ not computably isomorphic to a punctually 1-decidable copy.*

Proof. Let $\{\langle \varphi_e, L_e, A_e\rangle \mid e \in \omega\}$ list all partial computable functions together with all punctually 1-decidable linear orderings, where $A_e(x, y)$ either punctually yields T or gives z with $x <_{L_e} z <_{L_e} y$. (That is, or course, we are listing a primitive recursive function enumerating L_e and one generating A_e.)

We build $\hat{L}$ as $\omega^* + O_1 + O_2 + \dots$ where the ordering O_e is devoted to the e-triple. The sections O_e work independently. For O_e we begin with a pair $[a, b]$ and start to build an $\omega + \omega^*$ sequence, by making $a < a_1 < a_2 < \dots a_s b_s < b_{s-1} < \cdots < b_1 < b$, at stage s so that $[a_s, b_s]$ is the single stage s adjacency not yet declared as an adjacency in $\hat{L}$.

If this continue forever, clearly we will build $\omega + \omega^*$ with endpoints a and b. We are waiting for $\varphi(a)\downarrow, \varphi(b)\downarrow [s]$. Now if s occurs, before we move to stage $s + 1$, we will now invoke L_e and A_e sequentially on $[\varphi(a), \varphi(b)]$ and then on elements produced inside of this interval by L_e and A_e, to see whether $\left|[\varphi(a), \varphi(b)]_{L_e}\right| > 2s + 3$. If the answer is yes, then we add a single element d_s between a_s and b_s and declare that $[a_s, d_s]$, $[d_s, b_s]$ to be adjacencies. Note this diagonalizes the e-th triple. In the case that $\left|[\varphi(a), \varphi(b)]_{L_e}\right| \leq 2s+3$, then we will simply decide to make $[a, b]$ an $\omega + \omega^*$ sequence, again diagonalizing e.

Note that if we choose the first option, then the outcome is that $[a, b]$ is finite and will be absorbed by the next pair. Since there are infinitely many pairs with the second outcome, the final order type is of the desired form. □

The above can be extended to show that punctual 1-decidability and 1-decidability talk about distinct classes of orderings.

Theorem 2.8. *There is a 1-decidable linear ordering $\hat{L}$ which is not isomorphic to a punctually 1-decidable one.*

Proof. This time we build $\hat{L}$ to meet the requirements:

$$R_e : \hat{L} \ncong \langle L_e, A_e \rangle$$

Here $\langle L_e, A_e \rangle$ are as in Theorem 2.7. We will use an analog of the Jockusch-Soare idea of *separators.* ([41])

The ordering we build will be of the form $S_1 + O_1 + S_2 + O_2 + S_3 + \dots$, where S_i is of the form $2 + \mathbb{Q} + 2i + 1 + \mathbb{Q} + 2$. In the orderings O_i we will never have a complete block with odd number of adjacencies, so the separators in some L_e, if present, are completely determined by a 6-tuple, $x_1, \dots, x_6$, with x_1, x_2 and x_5, x_6 adjacencies, x_3, x_4 the block of $2i + 1$ elements. In any L_e, A_e, note that $x_1, \dots, x_6$ being a separator is a Σ^0_2 event, as the successivities are primitive recursive. At and stage we can believe that at most one 12-tuple codes a pair of consecutive separators, we can stick to believing it unless it proves wrong at some stage; and if this ever proves wrong then that tuple never again looks correct. We can have a list of possible 12-tuples coding separators corresponding to S_e and S_{e+1} in L_e, A_e.

For the sake of R_e we work in O_e. In $\hat{L}$ our copy of the separator e and $e+1$, will be given as $(a_1, \dots, a_6), (b_1, \dots, b_6)$, and O_e is built between a_6 and b_1. At every stage we will densify the regions $[a_2, a_3], [a_4, a_5], [b_2, b_3], [b_4, b_5]$.

The strategy is as follows. Initially, we will begin to build an ω-sequence $a_6, c_1, c_2, c_3, \dots c_{2n} b_1$, above a_6, and making b_1 a left limit point if nothing else happens. At every stage, we have declared every $[c_i, c_{i+1}]$ and adjacency, as is $[a_6, c_1]$, with the last pair $[c_{2n}, b_1]$ declared a non-adjacency. Initially, we will have $k^s_e = b_1$.

We let R_e require attention at stage n, if $\langle L_e, A_e \rangle$ has a 12-tuple $(x_1, \dots, x_6)(y_1, \dots, y_6)$ which appears to be correspond to the separators S_e, S_{e+1}. Now at such a stage, before we move to stage $n + 1$, we will run the enumeration of L_e, A_e a primitive recursive number of steps and examine the ordering $[x_6, y_1]$ in L_e. There are several cases.

(1) There are $\leq 2n + 4$ many points in $[x_6, y_1)$ in L_e (and hence all are adjacencies).

   *A*ction. In this case we simply continue to add two new points c_{2n+1}, c_{2n+2} to the end of the sequence, declaring $[c_{2n}, c_{2n+1}]$ and $[c_{2n+1}, c_{2n+2}]$ as adjacencies and $[c_{2n+2}, b_1]$ a non-adjacency, as usual. Note that after we add two new points in the next stage, R_e will be met if, indeed, the 12-tuple $(x_1, \dots, x_6)(y_1, \dots, y_6)$ is correct. Continue to keep $k^{n+1}_e = k^n_e = b_1$.

(2) There are more than $2n + 4$ many points $x_6, p_1, \dots, p_q$ in $[x_6, y_1)$ in L_e In this case, let $d_{e,n}$ be the point of $L_e[n]$ immediately left of y_1. Note that $d_{e,n} \geq p_{2n+4}$ in L_e.

   Subcase 1. The points in L_e $p_1, \dots, p_{2n+4}$ form consist of adjacencies $[x_6, p_1], [p_1, p_2], \dots [p_{2n+3}, p_{2n+4}]$.

   Action. Define $h^s_e = c_{2n+2}$ and in $\hat{L}$ we will now begin to build an ω^* sequence with right limit point h^s_e.

   Note that if L_e has correctly identified $(x_1, \dots, x_6)(y_1, \dots, y_6)$, then we will build between a_6 and b_1 and ordering of type $(2n + 2) + \omega^*$ whereas whatever L_e tries to build, he has told us that there are at least $2n+4$ adjacencies immediately right of x_6. So we diagonalize L_e, A_e.

Subcase 2. He has not yet given us at least $2n+4$ adjacencies immediately left of x_6. Let $d_e^n = p_q$, the point of L_e immediately left of y_1 at this stage n. Note that there are more that $2n+4$ points left of $d_{e,n}$ and above x_6.

Action. In this case we will move to the next stage and continue to monitor the situation, each time continuing to build our ω-sequence with limit $k_e^s = b_1$. The key observation is that, at each stage $n' > n$ if he ever includes $d_{e,n}$ into an initial segment of adjacencies we can diagonalize via Subcase 1. That is, for example, at stage $n+1$ we will have constructed $c_1, \ldots, c_{2n+2}$ as adjacencies. We will then look at L_e, A_e and see if $d_{e,s}$ is included in an initial segment of adjacencies from x_1. If it is, then since this must have at least $2n+4$ elements we can switch back to building an ω^* sequence, setting $c_{2n+2} = h_e^s$. Moreover, at this stage, we will also be similarly be able to diagonalize of there is an initial segment of adjacencies of size at least $2n+4$, so, inductively, we can conclude that

- d_e^n will never be part of an initial segment of adjacencies from x_6.
- We build a sequence of type ω with limit point b_1.

That means that we diagonalize $\langle L_e, A_e \rangle$, because he has some element $d_e^n < y_1$ with infinitely many elements left of it in the interval $[x_6, y_1]$, and we don't.

The conclusion is that after this attack, either we will will be building a sequence of type $h_e^n + \omega^*$ or one of type $\omega + 1$.

Suppose that later we see that our 12-tuple is wrong. By induction we can assume that we are building a sequence of type $h_e^n + \omega^*$ or one of type $\omega + k_e^n$. Once the 12-tuple is wrong, within the interval we would go back to building a sequence of type $\omega + k_e^n$. Now when a new 12-tuple $(x_1', \ldots, x_6')(y_1', \ldots, y_6')$ appears correct, there will be analogous cardinality questions. In general we'll be at some stage where we will have constructed $a_6, c_1, \ldots, c_{2s}$ immediately right of a block of size $k_e^s = k_e^n$, with $[c_{2s}, k_e^s]$ declared a non-adjacency. We need to do nothing until a stage is reached where the last k_e^s many elements form a block in L_e. Assume that this already the case. We would examine $\langle L_e, A_e \rangle$ to see if there are more than $2s + 4 + k_e^s$ many elements in this interval in L_e between the separators. Note that if not we win but continuing as we have done. If there are, then either we can identify an element d_e^s which will be left of the final k_e^s-block and has infinitely many elements left of it in $[x_6', y_1']$ and our corresponding order type is $\omega + k_e^s$, or at some stage he gives us too many adjacencies right of x_6', and we then diagonalize via an order type $h_e^t + \omega^*$.

If he switches 12-tuples infinitely often he loses. Thus we have shown that $S_e + O_e + S_{e+1}$ diagonalizes $\langle L_e, A_e \rangle$. □

Many results about linear orderings have analogs for boolean algebras, often because of the following representation theorem.

Theorem 2.9 (The Representation Theorem, folklore after Stone)**.** *Every countable boolean algebra is isomorphic to an interval algebra $Intalg(L)$ of a linear ordering of the same degree. (Recall that $Intalg(L)$ is the algebra generated by the finite unions of left closed right open intervals of the ordering L.)*

This first result is proven by effectivizing Stone's proof. Of course, this also shows that every computable boolean algebra is a computable subalgebra of $\mathbb{Q}$, here denoting the free boolean algebra (this will be clear from context). Theorem 2.8 is, however, not true for boolean algebras. We need the following well-known result.

Theorem 2.10 (Remmel-Vaught [61]). [5]
Suppose that B is any boolean subalgebra of $\mathbb{Q}$ with infinitely many atoms $\{d_i : i \in \mathbb{N}\}$. Let $\widehat{B}$ be the subalgebra of $\mathbb{Q}$ obtained from B by splitting each atom of B a finite number of times in $\mathbb{Q}$ (together with B). (Thus the atoms of $\widehat{B}$ would be $\{e_{i_1}, ..., e_{i_{n(i)}} : i \in \mathbb{N}$ where $d_i = e_{i_1} \vee ... \vee e_{i_{n(i)}}\}$.) Then $\widehat{B}$ is isomorphic to B.

For proof of these two results, we refer the reader to, for example, Downey [22]. Note that, again, B is punctually 1-decidable iff the atom relation is punctually decidable with a function f which says that either T (if a is an atom) and, if a is not an atom, then $f(a)$ produces a splitting of a. Of course, atoms correspond to adjacencies in the interval representation of B. The following would be known to anyone who thought about it, and was first noted by Alaev, as part of a proof of more complex result (Theorem 2.14) with a different proof.

Theorem 2.11 (Alaev [2]). *If B is a 1-decidable boolean algebra, then B is isomorphic to a punctually 1-decidable boolean algebra.*

Proof. We start by being given B, a 1-decidable boolean subalgebra of $\mathbb{Q}$ and, since it is trivial otherwise, we assume B has in finitely many atoms, with a suitable algorithm A where $A(x)$ declares whether $a \in B$ is an atom. Thus at each stage s, we will have enumerated B_s, and are waiting for $t > s$ where, for $a \in B_s$, $A(a)$ to declare whether a is an atom of B. This must happen by some stage $t = t(a) > s$, and moreover if a is not an atom, then we can also wait for $a_1 \vee a_2 = a$ to occur in B_t, before allowing $A(a) \downarrow [t]$. To construct $\hat{B}$ we will follow B_s but continue to split $a \in \text{Atom}(B_s)$, initially as $a = e_1 \vee e_2$, and for stages $v > s$ until $A(a) \downarrow [t]$, continue to split the splittings of a. If $A(a)$ says that "s is an atom of B", then we will have, in $\hat{B}$, enumerated $d_1 \vee d_2 \vee \cdots \vee d_q = a$. At this stage we declare in $\hat{B}$ that $d_1, \ldots, d_q$ are atoms. If $A(a)$ says that $a = a_1 \vee a_2$, then we identify a_1 with e_1 and a_2 with e_2, and continue now with these each taking the same role as a.
The result now follows by Theorem 2.10. □

By a result of Tarski, the theory of boolean algebras admits effective elimination of quantifiers down to the atom relation. Thus, if B is 1-decidable, then in fact B is a decidable boolean algebra (e.g. Goncharov [35]). Since Tarski's decision procedure ([67]) is primitive recursive (see e.g. Doner and Hodges [21]) we get the following. corollary.

Corollary 2.12. *Every decidable boolean algebra is isomorphic to a punctually decidable one.*

There are a couple of further results about punctually 1-decidable linear orderings and boolean algebras.

[5]In his Thesis, Vaught proved this result for atomic boolean algebras and Remmel extended the result to all countable boolean algebras

Theorem 2.13 (Blinov [11]). *In the class of primitively recursive 1-decidable linear orderings, the ones which are primitively recursively categorical are exactly those with a finite number of adjacencies.*

We omit the proof of Blinov's Theorem. His result occurred after the analogous result for boolean algebras.

Theorem 2.14 (Alaev [2]).
(1) *A boolean algebra B has a punctually 1-decidable presentation iff it has a computable presentation with a computable set of atoms.*
(2) *A punctually 1-decidable boolean algebra is punctually categorical in the class of punctually 1-decidable boolean algebras iff it has a finite set of atoms.*

Proof. We give a new proof, using a technique along the lines of Theorem 2.11. For the hard direction, we are given a boolean algebra B which is 1-decidable and has infinitely many atoms, and we will suppose that A is a primitive recursive algorithm which either splits $a \in A$ or asserts a is an atom. We must build $C \cong B$ with C not primitive recursively isomorphic to B. The strategies work independently. Let $\{\phi_e \mid e \in \mathbb{N}\}$ list the primitive recursive functions. We work by diagonalization. for ϕ_0 we will enumerate B until A says some $a \in B_s$ is an atom. Whilst we are doing this, we will have C_s copying B_s. Now the only problem is that ϕ_e might be slower than the enumeration of B and the algorithm A. Thus what we will now do continue to split everything in C_t for $t > s$ until $\phi_0(a)\downarrow$ (so that it is a member of C_t) and split everything in C_t for one more step, $t+1$, and declare that this is t_0, the diagonalization stage for ϕ_0. ϕ_0 is diagonalized as $\phi_0(a)$ is a member of C_t and everything in C_t is not an atom. At stage $t+1$, we will declare any heir of a (i.e. splits of this in C_{t+1}) now to be atoms of C, and furthermore for any d which has occurred in $C_{t+1} - C_s$, which also has been split, and which A says is an an atom, we will declare the heirs of such d in C_{t+1} as atoms. We now move on to ϕ_1.

For ϕ_1 the method is similar but slightly more combinatorial. Now we ask for $t_0 + 1$ many atoms of $C_s - C_{t_0}$, say $a_1, \dots, a_{t_0+1}$ where in C_v as these new atoms appear in A_v, we will continue to split all of the elements of C_u where are not atoms of C_{t_0} for $u \geq v$ until $\phi_a(a_i)\downarrow [q]$ for all $1 \leq i \leq t_0 + 1$. This time we will split one further time, and again know we have diagonalized, as there are are more than $t_0 + 1$ many atoms of A_q and at least one must me mapped to a non-atom of C_{q+1}. Then we let $t_1 = q + 1$ and move on to the next requirement in the same manner.

The result then follows by Theorem 2.10. □

Whilst it is possible to define punctual 1-decidability for more general structures, there is as yet little further work.

Another approach to defining analogs of being strongly computable was defined by Bazhenov, Kalimullin, Melnikov and Ng [9] for finite functional languages. Let $\mathcal{A}$ be a punctual structure in a finite functional language. We say that $\mathcal{A}$ is *honestly generated* if there is a primitive recursive procedure which, for every term t and each element $\overline{a} \in \mathcal{A}$,

(1) decides whether $\exists \overline{x}\, (t(\overline{x} = \overline{a})$, and
(2) if the answer is "yes", it gives such an $\overline{x}$.

One of the result from [9] shows that being honestly generated still does not characterize the primitive recursive presentations.

Theorem 2.15 (Bazhenov et. al. [9]). *There is a finitely generated structure $\mathcal{A}$ with infinitely many non-punctually isomorphic punctual presentations and for which $\mathcal{A}$ has a unique (up to punctual isomorphism) honestly generated punctual presentation.*

3. Online structures II

In [23], Downey, Melnikov and Ng observed that the punctual model did not capture all of the intuition of online combinatorics. Consider online colouring a graph. In the simplest case, we would be given the graph $G = \lim_s G_s$, where G_s has s vertices. When the vertex s is introduced, we are also given at the same time precisely which vertices amongst $\{1, \ldots, s-1\}$ has an edge with s (and this cannot change later). (This is the "request set" in Karp's paper.) Our task is to colour s so that no two vertices which are connected have the same colour, before the opponent presents us with G_{s+1}.

Although in practice the task will be finite, since we have no idea how large the graph is, we can construe this as an infinite process. Imagine this online colouring of a finite graph as an infinite process where we need to colour the whole of an infinite graph G given to us as incremental induced subgraphs. We can think of each possible version of G as being a path through an infinite tree of possibilities. Each node σ of length s of the tree will represent some graph. Each node σ of length s of the tree will represent some graph G_σ with s vertices, and if $\sigma \prec \sigma'$ then G_σ is the subgraph of $G_{\sigma'}$ induced by vertices $\{1, \ldots, s\}$. Note that there are only primitively recursively many non-isomorphic graphs with s vertices[6]. Then this view of an online algorithm differs from that given in [7] for the following core reason:

> Although G can be viewed a path on an infinite primitive recursive tree of possibilities, there is no *a priori* reason that we should only consider a primitive recursive graph G. There are continuum many such paths and the online graph colouring problem can be considered for an infinite countable graph of any complexity.

Thus the area becomes one where we are looking at punctual computable operators acting *uniformly* (and *continuously*) on trees of possibilities. A *filtration* (terminology of [23]) or *online presentation* is a path through the tree of possibilities given stage by stage. The area becomes a subarea of *computable analysis*, and is a "type II" theory[7].

The reader will note that an online presentation of a structure G_α is $\lim_{\sigma \prec \alpha} G_\sigma$ where $|\sigma| = s$ will be quite sensitive to coding. For example, any computable model could be slowed down, so what is achieved? However, any *natural* online structure arising in practice has a *natural* online presentation, and we see this as a feature rather than a deficiency.

[6]Meaning that this number is $u(s)$, where u is primitive recursive.

[7]Here "type II" refers to the fact that we are concerned with computable *functionals*, this terminology going back to work of Kleene in the 1950's. The connection between finite combinatorics and analysis is well-known especially in number theory, and a modern formal example is Avigad [6]. The stress here is on the *uniformity* of the finite combinatorics. For example, we know that for planar graphs given offline, we can 4-colour them in polynomial time. However, if we are given a huge planar graph where we cannot know the graph as a whole, we are essentially in an online situation and we know the situation becomes algorithmically hopeless

For the present, we consider only relational structures with a finite signature.

Definition 3.1. A online problem is a triple (I, S, s), where I is the space of inputs (i.e. the filtration) viewed as finite strings in a finite or infinite computable alphabet, S is the space of outputs viewed as finite strings in (perhaps, some other) alphabet, and $s : I \rightrightarrows S^{<\omega}$ is a (multi-)function which maps each $\sigma \in I$ to the set $s(\sigma)$ of admissible solutions of σ in S.

For instance, for a colouring problem I will be codes for finite graphs and S for finite coloured graphs. Then $s(\sigma)$ will correspond to the collection of all admissible colourings, e.g. such that adjacent vertices are distinctly coloured. These colourings will form the space of admissible solutions.

Definition 3.2. A punctual solution to (a representation of) an online problem (I, S, s) is a computable function $f : I \to S$ with the properties:

(O1) $f(\sigma) \in s(\sigma)$ for every $\sigma \in I$;
(O2) If $\sigma \prec \tau$ then $f(\sigma) \preceq f(\tau)$;
(O3) f is primitive recursive. (Moreover $f(\sigma)$ halts before any $f(\sigma')$ for $\sigma \prec \sigma'$).

One natural example of an online problem in this general sense is the construction of computable martingales betting on $\alpha \in 2^{\omega}$. m bets on $\sigma 0$ and $\sigma 1$ using the capital $m(\sigma)$. The set of "solutions" are the possible values for $m(\sigma)$, which can be calculated in advance.

The reader should note that several of the theorems we have already met do translate into the "operator online" setting. This is particularly true of results parameterized by some structural metric. Without giving as general definition, Downey and McCartin [28, 30] introduced online parameterized complexity analysis especially in the context of parameterized approximation for graphs, and partial orderings. (See Downey and Fellows [25, Chapter 31.4].) For example, the *proof* of Theorem 1.4 also shows.

Theorem 3.3. [8] *Every online graph G of pathwidth k can be online coloured with at most $3k + 1$ many colours.*

There has been a lot of work on colouring interval graphs online. One line of attack is an analysis of the dumbest method of colouring possible: First Fit. The first work here was by Witsenhausen [70], and independently Gyárfás and Lehel [39], where they obtain upper bounds of the form $c(\epsilon)(k+1)^{1+\epsilon}$, for an interval graph of width k. Gyárfás and Lehel [39] also proved an upper bound of $c(k+1)\log(k+1)$. The first result showing the linearity of First Fit was Kierstead's 1988 proof [47] that first fit will colour interval graphs of width k in at most $40(k+1)$ many colours. In 1995, Kierstead and Qin [49] improved this approximation ratio to 26. Finally, in 2008, Naranaswamy and Babu [59] gave an approximation ratio of 8. We remark that in 1988, Chrobak and Ślusarak [18] gave a lower bound of 4.4. And in 2016 Kierstead, Smith, and Trotter [50] showed that the performance ratio of first fit on interval graphs is at least 5. The precise best bound is not yet known. It is natural to ask whether, like Kierstead and Trotter's results on interval graphs, results of this ilk can be extended to graphs of bounded pathwidth. It is not clear however that any of the interval graph results transfer to graphs of bounded pathwidth.

[8]Again, this was proven for interval graphs by Keirstead and Trotter [54].

This is because the proofs of the upper bounds use properties exclusive to interval graphs (and k-paths).

All proofs of upper bounds for first fit use a similar analysis. Since we know the algorithm, namely first fit, what is done is to examine the effect that this algorithm will have on an interval representation of the structure. This method can be adapted to graphs of bounded pathwidth, as we now see, although we dont yet know how to get a linear upper bound. To explain this method we begin with a simple upper bound for graphs of bounded pathwidth. We first show that First Fit's competitive ratio for graphs of bounded pathwidth k is at most $(k+1)^2$. The proof is based off a rather crude analysis, and after this we show that a more careful analysis gives a better bound. However this proof serves to show that the competitive ratio is bounded for any fixed k.

Theorem 3.4. *First-Fit requires at most $(k+1)^2$ colours to colour an online graph with pathwidth k.*

Proof. Let G be a graph with pathwidth k. Suppose that G has been coloured using First-Fit. Let c be the largest colour used by First-Fit, and v a vertex coloured c. Fix $\mathcal{P} = \{P_1, P_2, \dots\}$ a path decomposition of G. For a vertex w let the *thread* of w be $\mathcal{Q}_w = \{P_i \in \mathcal{P} : w \in P_i\}$, the set of all bags from $\mathcal{P}$ containing w.

In stages we will construct a sequence of vertices $w_1, \dots, w_{k+1}$ such that all the bags in $\mathcal{Q}_{w_{i+1}}$ are contained in $\mathcal{Q}_{w_i}$. Denote by $f(n)$ the colour assigned to w_n.

Stage 1: Let $w_1 = v$. Hence $f(1) = c$.

Stage 2: The vertex $w_1 = v$ has colour c and thus is connected to $c-1$ vertices $U = \{u_1, \dots, u_{c-1}\}$ where each u_i is coloured i. Let a (and b) be the smallest (and largest) such that the bag P_a (and P_b) contains v. That is, P_a and P_b are the end points of the thread $\mathcal{Q}_v$. The bags P_a and P_b both contain v and have size at most $k+1$. Thus P_a and P_b contain at most k vertices from U. Hence there are at least $c-1-2k$ many u_i's not in P_a and P_b. These vertices must be in bags between P_a and P_b. Because these u_i's are not in P_a or P_b they cannot be in any bags outside P_a and P_b. Let w_2 be one of these middle vertices with colour at least $c-1-2k$.

In summary, if the bags P_a and P_b are 'full', then the remaining u_i vertices are trapped between P_a and P_b. We can then pick one with a sufficiently large colour.

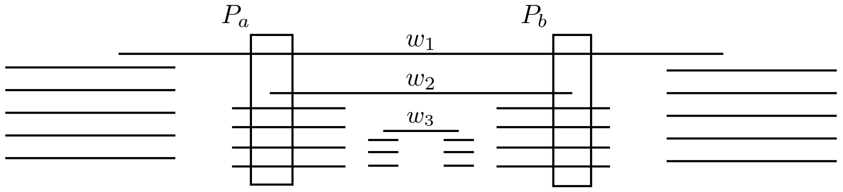

FIGURE 1. A Diagram showing the threads (horizontal lines) for w_1, w_2, w_3 along with the bags P_a and P_b for stage 3

Stage n+1: As w_n is coloured $f(n)$, the vertex w_n must be connected to vertices $U = \{u_1, \dots, u_{f(n)-1}\}$ where each u_i has been coloured i. Let a (and b) be the smallest (and largest) such that the bag P_a (and P_b) contains w_n. By construction the bags P_a and P_b contain the vertices $w_1, \dots, w_n$ and have size at most $k+1$. Thus P_a and P_b each contain at most $k+1-n$ vertices from U. The remaining $f(n)-1-2(k-n+1)$ many u_i's not in P_a and P_b are in bags between P_a and P_b.

That is $|U \setminus (P_a \cup P_b)| \geq f(n) - 1 - 2(k - n + 1)$. Further these u_i's cannot be in any bags outside P_a and P_b as they are not in P_a or P_b. Let w_{n+1} be one of these vertices from $U \setminus (P_a \cup P_b)$ with colour $f(n+1) \geq f(n) - 1 - 2(k + 1 - n)$.

This ends the construction. We have $f(1) = c$ and $f(n+1) \geq f(n) - 1 - 2(k - n + 2)$. The solution to this recurrence relation is $f(n) = -2k(n-1) + n(n-4) + c + 3$.

The vertex w_{k+1} has thread $\mathcal{Q}_{w_{k+1}}$. But every bag in $\mathcal{Q}_{w_{k+1}}$ contains the vertices $w_1, \ldots, w_{k+1}$. Thus w_{k+1} cannot be adjacent to any colour $\leq f(k)$ and must have colour 1. That is $f(k+1) = 1$. Hence we have $1 = f(k+1) \geq -k^2 - 2k + c$. Therefore $c \leq k^2 + 2k + 1$, as desired. □

Note also that the proof will also transfer back to computable graph theory:

Corollary 3.5. *First-Fit requires at most $(k+1)^2$ colours to colour a computable graph with pathwidth k.*

Our better bound is based on a method due to W. Just in [39], which we modify from interval graphs to graphs of bounded pathwidth. We will let $\chi_{\mathrm{FF}}(G)$ denote the first fit colouring number of G.

Again, for G a graph let $\mathcal{P}_G$ be a particular path decomposition of G with minimum width. For a vertex v in $V(G)$ let $\mathcal{Q}_v$ be the thread of v in $\mathcal{P}_G$.

For $P \in \mathcal{P}_G$ and $v \in V(G)$ we define the following.

- $\rho(P, \mathcal{P}_G) = |\{v : v \in G, P \in \mathcal{Q}_v\}|$, The number of vertices that contain P in their thread. Or, the size of P.
- $\rho(v, \mathcal{P}_G) = \min_{P \in \mathcal{Q}_v} \rho(P, \mathcal{P}_G)$, The smallest bag in v's thread.
- $\rho(\mathcal{P}_G) = \max_{v \in V(G)} \rho(v, \mathcal{P}_G)$, The largest smallest bag in every thread.
- $f(k) = \max\{\chi_{\mathrm{FF}}(G) : \mathrm{pw}(G) = k - 1\}$
- $g(k) = \max\{\chi_{\mathrm{FF}}(G) : \rho(\mathcal{P}_G) = k\}$

Note that we have, $\mathrm{pw}(G) + 1 = \max_{P \in \mathcal{P}_G} \rho(P, \mathcal{P}_G)$. We also have $\mathrm{pw}(G) + 1 \geq \rho(\mathcal{P}_G)$ and so $g(k) \geq f(k)$ follows from the definition.

We will treat a colouring of a graph G by First Fit using m colours as a partition $V(G) = \mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \cdots \cup \mathcal{I}_m$. Each $\mathcal{I}_i$ is an independent set and for all $v \in \mathcal{I}_i$ and $j < i$ there is a vertex $u \in \mathcal{I}_j$ such that u and v are adjacent. We may write $\mathcal{P}_{\mathcal{I}_i}$ as shorthand for the path decomposition that is a subset $\mathcal{P}_G$ and contains only the vertices in $\mathcal{I}_i$.

Lemma 3.6. *For every graph G, we have $\rho(\mathcal{P}_G) \geq \lceil (\mathrm{pw}(G) + 1)/2 \rceil$*

Proof. Let $k = \mathrm{pw}(G)$. Fix some $P = \{v_1, \ldots, v_{k+1}\}$ a bag in $\mathcal{P}_G$ of size $k+1$. Let A be the first $\lfloor k/2 \rfloor$ vertices chosen by increasing left endpoint of their threads. Let B be the first $\lfloor k/2 \rfloor$ vertices chosen by increasing right endpoint of their threads.

The size of $|A \cup B|$ is at most $2\lfloor k/2 \rfloor$. The remaining vertices in $P \setminus (A \cup B)$ have threads contained entirely within the threads of the vertices in $A \cup B$. Thus for each $u \in P \setminus (A \cup B)$ the thread $\mathcal{Q}_v$ is contained in $\lfloor k/2 \rfloor$ other threads. Therefore $\rho(\mathcal{P}_g) \geq \rho(u, \mathcal{P}_P) \geq \lfloor k/2 \rfloor + 1 = \lceil (k+1)/2 \rceil$. □

Theorem 3.7.

$$g(2k) \leq 2g(k) + 6k - 2$$

Proof. Assume for a contradiction that $g(2k) \geq 2g(k) + 6k - 1 = n$. Then there exists a graph G with $\rho(\mathcal{P}_G) = 2k$ such that First Fit partitions G into the sets $\mathcal{I}_1, \ldots, \mathcal{I}_n$.

Consider the set $\mathcal{I}' = \bigcup_{i=m}^{n} \mathcal{I}_i$ where $m = g(k) + 6k - 1$. We can see that $\mathcal{I}'$ is partitioned into $n - m + 1 = (2g(k) + 6k - 1) - (g(k) + 6k - 1) + 1 = g(k) + 1$ parts by First Fit. Therefore by the definition of $g(k)$ there exists some vertex $v \in \mathcal{I}'$ such that $\rho(v, \mathcal{P}_{\mathcal{I}'}) \geq k + 1$.

Define A as the set of all elements j from $\{1, \dots, m-1\}$ such that $\mathcal{I}_j$ contains a vertex u such that u is in an end bag of $\mathcal{Q}_v$. Let $B = \{1, \dots, m-1\} \setminus A$. For all $j \in B$ let $\mathcal{I}_j^*$ be a subset of $\mathcal{I}_j$ such that all vertices $u \in \mathcal{I}_j^*$ have $\mathcal{Q}_u \subset \mathcal{Q}_v$. As v is connected to a vertex from each $\mathcal{I}_j$ where $j \leq m - 1$ the vertex v must be connected to a vertex w_l in each $\mathcal{I}_l$ where $l \in B$. Further w_l is not in the end bags of $\mathcal{Q}_v$ and so must have $\mathcal{Q}_{w_l} \subset \mathcal{Q}_v$. Thus each $\mathcal{I}_j^*$ is non-empty.

Let $\mathcal{I}^* = \bigcup_{j \in B} \mathcal{I}_j^*$. First fit partitioned $\mathcal{I}^*$ into $|B|$ parts. The maximum bag size is $2k$ and the threads of all the vertices in $\mathcal{I}^*$ are subsets of $\mathcal{Q}_v$. Recall that $\rho(v, \mathcal{P}_{\mathcal{I}'}) \geq k + 1$. Hence $\rho(\mathcal{I}^*) \leq 2k - \rho(v, \mathcal{P}_{\mathcal{I}'}) \leq k - 1$. Thus we have $|B| < g(k)$. Therefore $|A| = m - 1 - |B| \geq 6k - 1$.

This shows that at least $3k$ vertices from $\mathcal{I} \setminus \mathcal{I}'$ are in one end bag of $\mathcal{Q}_v$. We also that $\rho(v, \mathcal{P}_{\mathcal{I}'}) \geq k + 1$, and hence every bag in $\mathcal{Q}_v$ contains $k + 1$ vertices from $\mathcal{I}'$. Thus one end bags of $\mathcal{Q}_v$ contains $4k + 1$ vertices. Finally by lemma 3.6 $\rho(\mathcal{P}_G) \geq 2k + 1$. This contradicts the fact that $\rho(\mathcal{P}_G) = 2k$. □

Theorem 3.8. *If G is an online or a computable graph with pathwidth K, $\chi_{\text{FF}}(G) \leq 3(k+1)\log_2(k+1) + 2 - (k+1)$.*

We make the following conjecture[9].

Conjecture 3.9. *First Fit will colour graphs of bounded pathwidth with a linear approximation ratio.*

One question is whether we can obtain better bounds for computable graphs than we can for online ones.

We remark that there is a dislocation between the punctual structure theory and the type 2 online theory we have described above. One good example concerns online categoricity. A punctual structure $\mathcal{A}$ is punctually categorical iff all punctual copies A_1, A_2 are punctually isomorphic (meaning there is a punctual isomorphism $f : A_1 \to A_2$, with a punctual inverse). The analogous definition for the general model of online structures is the following.

Definition 3.10. A structure G is strictly online categorical if there is an online strict operator f which, on input α and β arbitrary representations of G outputs an isomorphism from α onto β.

By strict, we mean that there exists a primitive recursive functional $f^{\alpha;\beta}$ with both uses being the identity function, such that the associated function $h(i) = f^{\alpha\restriction i;\beta\restriction i}$ (whose output is interpreted as a natural number) induces an isomorphism from α onto β; recall the latter two are isomorphic copies of G upon the domain $\mathbb{N}$.

Recall that a structure G is homogeneous if for any tuple $\bar{x}$ in G and any pair of elements $y, z \in G$, we have that y is automorphic to z over $\bar{x}$.

[9]Whilst the paper was in proof we became aware of the paper "An improved Bound for First-Fit on Posets without Two Long Incomparable Chains" by Vida Dujmovic, Gwenael Joret and David Wood (SIAM J. Discrete Math, Vol. 26 (2012), 1068-1075), who have shown that first-fit will colour graphs of pathwidth k with at most $8(k+1)$ many colours. The technique for the analysis is a static analysis of chain decompositions of posets.

Theorem 3.11 (Downey, Melnikov, Ng [23])**.** *A structure in a finite relational language is strictly online categorical if, and only if, it is homogeneous.*

Proof. Each homogeneous structure is trivially strictly online categorical. Now suppose G is strictly online categorical. Suppose the structure is not homogeneous, and let $\bar{x}$ be shortest (of length n) such that for some z, y we have that z is not in the same automorphism orbit as y over $\bar{x}$. Construct α and β as follows. First, copy $\bar{x}$ into both and calculate the online isomorphism f from $\alpha \restriction n$ to $\beta \restriction n$. If we identify $\alpha \restriction n$ and $\beta \restriction n$ with $\bar{x}$, then f induces a permutation of $\beta \restriction n$; by the choice of n any permutation of $\bar{x}$ can be extended to an automorphism of the whole structure. Adjoin z to α and find a y' which plays the role of y over $\beta \restriction n$ under any automorphism extending the permutation $\beta \restriction n \leftrightarrow f(\alpha \restriction n)$. Then necessarily $f(z) = f(y')$, because f has already shown its computation on the first n bits. However, by the choice of z and y', f cannot be extended to an isomorphism no matter how we extend the presentations further. $\square$

Note that we used only totality of the strict functional in the proof. In the case when the language has functional symbols the theorem no longer holds. Of course, the notion of strictly online and of a presentation will have to be adjusted. But regardless, strong homogeneity will no longer capture the property (whatever it may be exactly). The following example is from [23].

Example 3.12. Consider the structure in the language of only one unary functional symbol s, and which consists entirely of disjoint 2-cycles. Here a 2-cycle is of course a component of the form $\{x, s(x)\}$ where $s(s(x)) = x$ and $x \neq s(x)$. According to any reasonable definition of (strong) online categoricity for functional structures, this structure has to be (strongly) online categorical. However, it is not homogeneous.

Problem 3.13 (Downey, Melnikov, and Ng [23])**.** Is it possible to find a reasonable algebraic description of (strongly) online categorical algebraic structures in an arbitrary finite language?

This subject is awaiting development. One of the things which is lacking is a model theory. Finite structures have a well-developed finite model theory (See, for example, Libkin [56] or Grädel et. al. [36]). Two of the ideas are to use finite structures to capture complexity, and another is to obtain results about the algorithmics based purely on logical description. A quintessential result here is Courcelle's Theorem which concerns graphs of bounded treewidth. The concept of treewidth is the same as pathwidth except that in the place of a path with bags at the vertices, we have a tree. This is a measure of how "treelike" a graph is. In offline graph theory, being treelike seems to make computational questions more tame. Recall that the monadic second order MS_2 is a fragment of second order logic for graphs logic of graphs is a two sorted logic with vertex and edge variables, (represented by lower case variables) and variables for sets of vertices and edges (upper case), along with relations of the form $v \in e$, and allowing quantification over vertex and edge variables, and quantification over sets of variables also. For example, to say that G is 3-colourable we could write this as "there are three sets of vertices, disjoint, all vertices lie in exactly one, and if xy is an edge then both x and y are not in the same set of the partition."

Theorem 3.14 (Courcelle's Theorem [19], but see [25])**.** *Let C be the class of graphs of treewidth k. Any* MS_2 *formula is decidable in linear time for graphs in C. Moreover, given a minimization problem P which is* MS_2 *definable, then we can compute an optimal solution to P for members of C in polynomial time.*

We remark that this result gives us a very high level classification tool for algorithmic properties, and indeed some properties were shown to have polynomial time algorithms. There have been many extensions to Courcelle's Theorem for other classes of structures and fragments. For example, graphs of bounded "cliquewidth" have linear time algorithms for deciding first order formulae. (See, for example, Downey and Thilikos [29].)

We wonder if there are similar theorems available in the online situation. This is wide open. It is not even clear what logic would be correct.

A simple result along these lines is the following.

Theorem 3.15 (Downey and Long Qian-unpublished)**.** *Given any first order formula $\phi(X)$ in the language of graphs where X only occurs positively or only occurs negatively in $\phi(X)$, then the online problem corresponding to $\phi(X)$ (i.e. the maximization problem if X occurs negatively) has an online algorithm of constant competitive ratio on graphs of bounded degree. In fact, the greedy algorithm is sufficient.*

Proof. Without loss of generality, assume that X only occurs negatively in $\phi(X)$. Then $\phi(X)$ induces an online maximization problem attempting to maximize X. Consider the greedy algorithm where for each vertex presented, it's greedily added to X if possible. Suppose that a solution set of X' is built and X_{opt} is the optimal solution set. For each $x \in X_{\text{opt}}$, consider $\phi(X' \cup \{x\})$. We need a notion from finite model theory ([33]). Given a first order formula $\psi(X)$ which is positive, Gaifman's Locality Theorem ([33]) says that it is equivalent to a boolean combination of basic sentences of the form,

$$\exists x_1 \ldots \exists x_k \Big(\big(\textstyle\bigwedge_{1\le i<j\le k} \operatorname{dist}(x_i, x_j) > 2r \big) \wedge \big(\textstyle\bigwedge_{1\le i\le k} \xi(x_i) \big) \Big)$$

Here $\operatorname{dist}(x_i, x_j)$ refers to the distance between x_i and x_j (that is, the length of the shortest path from x_i to x_j), $\xi(X)$ is *r-local* meaning that every quantifier in it quantifies only about vertices in within a distance r of X. Since both $\phi(X')$ and $\phi(X' \cup \{x\})$ hold and $\phi(X)$ is r-local, $\phi(X' \cup \{x\})$ can fail to hold only if $X' \cap N_r(x) \neq \emptyset$, which is to say that x is in the r-neighbourhood of some element in X'. Therefore, for any node v added to X', at most $|N_r(v)|$ vertices from the optimal solution cannot be added to X'. This shows that the greedy algorithm will have a competitive ratio of $O(\max\{|N_r(v)|\}$. But the graph has bounded degree and hence this is a constant ratio. □

A trivial application of this metatheorem is Minimum Vertex Cover, which asks for the smallest set of vertices X such that for all $xy \in E(G)$, either $x \in X$ or $y \in X$, on graphs of bounded degree.

To speculate at this point, since the subject lies under the umbrella of computable analysis, it could be possible that the correct logic here might be a suitably decomposed analog of *continuous logic.* This is a logic where formulae are considered as functions and the truth values are reals between 0 and 1. Nothing has been done in this area and we refer the reader to Chang and Kiesler [17], and Ben Yaacov et. al. [72] for the classical versions. This would be the correct logic for numerical analysis.

4. (Online) Adversarial Algorithmic

4.1. **Adversarial.** A related area of computable combinatorics concerns (offline) problems of the following kind. Given a planar graph G we wish to colour it but have the help of a possibly non-cooperative partner. Player I (i.e. we) colours the first vertex and the partner (II) colours the second. Can player I get away with only using 4 colours? The answer is no. There is a strategy for player II to block us and use more colours. The first paper in this area was Kierstead and Trotter [51], who showed that I had a strategy using at most 33 colours. Moreover their proof showed could be used to show that for any finite genus graphs of that genus had an adversarial colouring number which was finite. It is only relatively recently this number was reduced to 17 in 2008 by Zhu [73]. It is unknown what the correct minimum number is.

For colouring more structured graphs, there is a strategy called the *activation strategy* which allows us to establish the bounds that I needs at most 4 colours for a tree (Kierstead [53]), for a graph of treewidth k, I needs at most $3k + 2$ colours (Wu and Zhu [71]), and Bodlaender showed in 1998 that graphs of pathwidth k I needs at most $3k + 1$ many colours. It seems reasonable to suggest that there is some relationship between the adversarial and the online versions.

As an example, we will consider how the activation strategy is used on trees. Fix some tree T. The first step of the activation strategy is to define a linear order L on $V(T)$ as follows. First we pick some vertex r to be the root of our tree. r becomes the least element in L. We then add all the children of r then all of their children and so on. That is we order $V(T)$ using *breath first traversal.* Player I then uses the activation strategy as follows.

(1) I starts by greedily colouring the least v in L.
(2) On I's next turn let u be the last vertex coloured by II. I starts at u activates it and moves to w, the least uncoloured neighbour of u in L (if u has no uncoloured neighbours then move to least uncoloured vertex in L).
(3) If w is activated or has no uncoloured neighbours then I greedily colours w. If not then I repeats step 2 on w until I either finds an active vertex that is activated or has no unmarked neighbours.

Fix some vertex uncolored v in $V(T)$. Because T is a tree there is exactly one neighbour of v that is less than v. I will never colour a bigger neighbour of v if v is unactivated. Hence I will not colour a second bigger neighbour of v before v. The second time a bigger neighbour of v is coloured by II v is coloured by I. Thus there are at most two coloured bigger neighbours of v when v is coloured. Hence there are at most three coloured neighbours of v that are coloured when v is coloured (the two bigger vertices and the smaller vertex). Therefore by following this method T can adversarial coloured in 4 colours.

It is clear that colouring is only one simple property that can be analysed in this setting. Any minimization property can have an adversarial version. For example, imagine a Dominating Set game of this type, were X is a dominating set if for vertices $v \in V(G)$, there is a vertex $x \in D$ such that $xv \in E(G)$. The smallest size of a dominating set is called the *dominating number.* Then every graph G with dominating number $\gamma(G)$ can be adversarial dominated with a dominating set of size $2\gamma(G) - 1$ (Brešar, Klavžar and Rall 2010 [14]). It is also conjectured that every forest with n vertices can be adversarial dominated with a dominating set of size $\lceil \frac{3n}{5} \rceil$ (Kinnersley, West, and Zamani 2013 [55]). While this conjecture is still undecided it has been shown for certain types of trees. For example, forests of pathwidth 1 graphs with no isolated vertices (Kinnersley, West, and Zamani 2013 [55]) and forests with no isolated vertices such that no two leaves have distance 4 (Bujtás, Csilla 2015 [15]).

4.2. **Online adversarial.** It is clear that both adversarial situations and online processes can be combined. That is, we are playing an online algorithm, but each second move the opponent gets a play. For example, in a colouring situation we'd have an unknown graph presented online, and where the opponent will colour each alternate vertex, say at odd stages. Now we would not have global knowledge of the object being constructed. This is not an unreasonable model to model interactions with the universe, which can be hostile.

There has been little work in this area and positive results seem difficult. For example, imagine a colouring scheme where our cost is not the cardinality of the set of colours we use, but the *number of* the colours we are forced to use; meaning that is there is a palate of colours $1, \ldots,$ then our cost is $\max\{n \mid$ we use colour $n\}$. Then with this model, if we are presented with n vertices which are disjoint, then the opponent can colour every second one a different colour and then, at step $n + 1$, we will be presented with a single vertex connected to them all. This forces us to use colour $\frac{n}{2} + 2$ at least to colour this vertex.

5. Random advice

We remark that a natural addition to this setting is either to consider online algorithms on random paths in some tree of possibilities, or to consider algorithms on online presentations with online advice at each step. Both of these could well take advantage of the work on algorithmic randomness as in Downey and Hirschfeldt [27], so that for e.g. Martin-Löf random paths the algorithm would work. For example, a random graph could be obtained by considering a computable bijection $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and then $\langle x, y\rangle$ is an edge iff $f(\langle x, y\rangle) = 1$ in the binary expansion of Chaitin's Ω. We won't pursue this here but it all look promising, as we would expect $0 - 1$-laws to be expressed.

Less well studied are algorithms with random advice, though there are some notable ones in finite combinatorics. For example, the Solovay-Strassen algorithm for primality. We will briefly look at one for finding paths in graphs. Here we are considering the product of space of online presentations and some copy of Cantor Space, so that on input G_σ for $|\sigma| = n$, when we add some new vertex v, we would toss a coin for advice (in Cantor Space) (typically a bit in $\{0, 1\}$, or some finite set) and the algorithm would act on $\langle G_\sigma \cup \{v_{n+1}\}, \tau * \langle i\rangle\rangle$ Here $|\tau| = n$ and represents one of the possible advice sequences, up to step n.

We will illustrate this approach with an online implementation of a famous algorithm for fining paths in graphs. Given a graph G with n vertices can we decide in polynomial time whether or not G contains a path of length $\log n$? This is known as the LOG PATH problem. It was conjectured by Papadimitriou and Yannakakis in [60] that LOG PATH is polynomial. This was answered in the affirmative by Alon, Yuster, and Zwick in [5].

Alon, Yuster, and Zwick introduced a randomised method for finding paths, cycles, and other small subgraphs called *color-coding*. To find a path of length k you randomly colour the graph using k colours. You then apply a dynamic programming approach to find a k-*colourful* path. That is a path of length k where every vertex is a unique colour. Using this method they prove the following.

Theorem 5.1. *Let $G = (V, E)$ be a graph and let $c : V \to \{1, \ldots, k\}$ a k-colouring of G. A colourful path of length $k-1$ can be found in G (if one exists) in $O(k2^k|E|)$ time.*

With this in mind, take some graph G with n vertices and a randomised $(\log n)$-colouring of G. As $\log n$ is fixed, we have a randomised algorithm to find a path of length $\log n$ (if one exists) in $O(\log n 2^{\log n}|E|) = O(n \log n|E|)$ time. Hence, LOG PATH is polynomial.

What if we do not know the number of vertices in the graph? In effect this would mean our graph is online. One simple method to find a path would be to re-run the randomised offline algorithm every time a new vertex is presented. Let $G = (V, E)$ be an online graph. At each stage s we run the randomised algorithm on G_s with s vertices, and look for a path of length $\log s$. At step $s+1$, we'd be tossing a k-sided die to determine the colour of the incoming vertex. This gives the online algorithm a complexity of

$$\sum_{s=1}^{n} s \log s|E_s| = O(n^2 \log n|E|)$$

The probability that a path in G is colourful is $k!/k^k > e^{-k}$, where k is the number of colours used. The randomised online algorithm may run multiple times looking for the same path of length $\log n$, and this can be done in parallel, with different advice strings. Hence, the probability that the online randomised algorithm finds a path of length $\log n$ is $\geq \log n/ \log n^{\log n}$. Therefore, the randomised online algorithm has the same (or better) probability of finding a path of length $\log n$ as the offline version.

If we want to derandomise this method, then we will need a list of k-colourings such that for every subset A of k vertices one colouring from the list makes A colourful. That is, we need a family of k-*perfect hash functions*. That is a set S of functions from $\{0, \ldots, n\}$ to $\{1, \ldots, k\}$, such that for each subset $D \subseteq \{1, \ldots, k\}$, there is a function $f \in S$ which is 1-1 on D. It turns out that the necessary size of such a family is $2^{O(k)} \log n$, for a graph with n vertices. (However, there is a problem as n needs to be large.) Therefore, we have a derandomised algorithm find a path of length k that runs in $2^{O(k)} \log nO(k2^k|V|)$ time.

While we cannot have a k-perfect hashing function on a graph with infinitely many vertices, we do not need to. Because we are applying the algorithm once at each stage, we can use a new family of hashing functions at each stage. This adds cost of $2^{O(\log s)} \cdot \log s = O(s \log s)$ to each stage, and so we have a running time of,

$$\sum_{s=1}^{n} s \log(s) s \log(s)|E_s| = O(n^3(\log n)^2|E|)$$

This last observation yields a new question, which is quite intriguing. The essence is whether you can modify the current family to include a new element, or need to run a new construction.

Question 1. What is the complexity of the following online problem, Incremental Hash Functions.
Input : A set S of k-perfect hash functions with domain $\{0, 1, \ldots, n\}$
Parameter : k
problem : Construct a set S' of k-prefect hash functions with domain $\{0, 1, \ldots, n+1\}$.

6. Strongly Online Algorithms and Functional Parameterizations

A new direction in the study of online algorithms takes inspiration from the material we have met on highly computable graphs. Consider the online problem of robot navigation in a maze. At any stage of the proceedings, the robot will have local knowledge of the maze. That is at location x in the maze (considered as a vertex) it will be aware of $N_1(x)$ the neighbours of x at distance 1; those it might choose as its next move. This is entirely analogous to the situation for highly computable graphs, since, knowing the valency of a vertex also gives us knowledge of the edges from the vertex. The idea here is that we are adding a new parameter to the descriptions, and this parameter is a function of the vertices in the part of the graphs our online algorithm has processed.

A stronger possible extension would be a strongly online presentation, where we add a function generating the neighbours of vertices in the language. Clearly in an online presentation, we would only be able to increment this at each step. This would be analogous to, for instance, the diffusion of something throughout a network, where our job was some kind of navigation within the network, which we are only slowly learning about. This idea may seem artificial to a graph theorist, but will be quite natural to an algebraist processing, say, an infinite finitely presented group on a laptop. Here, there are many algorithms which we know halt but have indeterminate running times, or at least unknown. Even in the situation of finite groups, if they are sufficiently large, algorithms often run with only local knowledge. (See, for example, [20, 65].) However, for such online situations, we have a function symbol in the language. At each stage we will be generating more and more of the structure, as we need to close under such functions. For example, we are running them for s many steps at step s. This make the situation more akin to a combination of type 1 (which stresses functions for constructions) and type II (where the main player seems to be relations).

In the first author's MSc Thesis [58], we formalized this in the following definitions.

Definition 6.1. Let $G = \lim_s G_s$ denote a presentation of an online graph.

(1) A *locally strongly online presentation* is a sequence $\langle G_s, H_s\rangle$ where $H_s = N[G_s]$ is the induced subgraph of G, G_s, together with its neighbours.
(2) A *strongly online* presentation of G is a pair $\langle G_s, H_s\rangle$ where this time $H_{s+1} = N[G_{s+1} \cup H_s]$.

The reader should think of G_s and the *blue* vertices and $H_s - G_s$ as the *red* vertices. The online algorithm will have the additional red information H_s but the algorithm only acts on the blue vertices, G_s. An example of a strongly online presentation problem is navigation where, whilst we are traversing some road network, we are gradually discovering the whole map in stages. Perhaps there is another agent whose job is to discover the nature of the network and report back to us. Whilst this might seem less natural, it seems worthy to study both kinds of online algorithmic parameterizations. [10] From the model-theoretical point of view, we can think of strongly online graphs as being ones where a function giving the neighbours of a vertex has been added to the language, and this extended structure has been enumerated gets enumerated by one step at each stage.

A more general approach for 1-decidable online structures would be to have the Σ_1^0 diagram decidable with primitive recursive witnesses at each step. This remains to be explored.

In the earlier sections, we have seen the interplay between online algorithms and computable graph theory (Kierstead [52]). In section 2, we have seen that in the type I primitive recursive interpretation of being online, if we can decide 1-quantifier statements, then, for example, we get varying categoricity results, as per Alaev [2], Blinov [11] and Bazhenov et. al. [9]. We do already know that parameterizations which impose shape restraints on the graph do give algorithmic advantages. Kierstead's survey [52] as well as Gasarch's [34] give ample evidence to this where almost every result relies on parametric considerations. We have seen in Theorem 1.4 and its online corollary, that graphs of pathwidth k can be coloured with with $3k-2$ many colours, and this is tight (Kierstead and Trotter [54]).

In this section we will concentrate on some new results giving consequences and limitations for (locally) strongly online graphs. All of the theorems proven in this section are due to the first author, Askes, under the direction of Downey as part of his MSc Thesis.

The following result shows that strongly online graphs tend to be reasonably tame for online colouring.

Theorem 6.2. *If G is a strongly online graph with chromatic number $\chi(G)$ then G can be online coloured with $2\chi(G)$ many colours.*

Whist we give a proof of theorem 1.6 in § 6.1 we remark that it essentially follows from an analogous theorem of Bean [10] for highly computable graphs[11].

It seems that being locally strongly online fails to give enough information some cases. For example, we have the following.

Theorem 6.3. *For every online algorithm A and $n \in \mathbb{N}$ there is a locally strongly online graph G that cannot be online coloured by A. Indeed, for forests of height n, the approximation ratio is $\Omega(\log n)$ which is no better than the one for normal online trees.*

We prove these theorems in § 6.1, where we also prove theorem 1.6 is tight. We also show that strongly online trees are 2-colourable.

In § 6.2 we will analyse colouring online graphs of bounded pathwidth. As with Kierstead's theorems for online graphs we get strong lower and upper bounds for strongly online graphs. We already know that strongly online graphs of pathwidth k are $2k+2$-colourable as they are $k+1$ offline colourable. We show:

[10]The reader might also think of this as adding a function to the language of the model and allowing bounded iterations at each stage. We have not explored the metatheory, yet.

[11]As we see, not every "highly computable" graph result has a strongly online analog, mainly because several algorithms on highly computable graphs have inherent delays, which are impossible in the online setting. For example, if we know that if T is a tree, then any algorithm can wait for local parts to be connected; something impossible in the online situation.

Theorem 6.4. *Every strongly online graph with even pathwidth k is strongly online $2k+1$-colourable.*

Theorem 6.5. *There is a strongly online graph of pathwidth k that is not strongly online $2k$-colourable.*

Theorem 6.6. *There is a strongly online graph of odd pathwidth k that is not strongly online $2k+1$colourable.*

In view of the focus on how the topology (i.e. "shape") affects the performance of strongly online graphs, it seems reasonable to suggest that we might be able to construct approximate path decompositions online, at least in the case of strongly online graphs. This would also seem somewhat of a breakthrough since current algorithms such as Bodlaender's [12] and Bodlaender-Kloks' [13] are highly recursive, and also use MS_2 definability. Unfortunately, as we see below, there is no hope of such an algorithm, even in the case of strongly online graphs. After we introduce a natural notion of an online path decomposition, we prove the following.

Theorem 6.7.
(1) *For each online algorithm A and finite number n, there is a strongly online (finite) G of pathwidth 2 which cannot have a online path decomposition of width n. We can construct G with at most $O(n^4)$ many vertices.*
(2) *There is an (infinite) strongly online graph of pathwidth 2 which cannot have a online path decomposition*

Finitizing this result, it shows that there is no d and online algorithm A, such that A takes a strongly online G of pathwidth 2 and produces an online path decomposition of width d.

When colouring strongly online graphs we can *assign* a colour to vertex in H_s, or we can irrevocably *colour* them. Because we must colour the vertices in the online presentation order, a simple strategy is to *assign* colours to vertices that we can see, but have not been presented. Henceforth, we will not make the distinction between assigning colours and colouring vertices. The assumption is that at each stage of a strongly online graph we will colour the vertex that must be coloured.

Likewise, when presenting/describing a strongly online graph and we do not specify which vertex to present next, it can be assumed that we are presenting a vertex in V_{s-1}. This vertex has been revealed and it's neighbours are in the next boundary ($N(V_{s-1})$).

For this section, let $G = \lim G_s$ be an online graph where $G_s = (V_s, E_s)$. We denote $\hat{V}_{s+1} = V_{s+1} \setminus V_s$, and $\hat{E}_{s+1} = E_{s+1} \setminus E_s$. Similarly $\hat{G}_s$ is the induced subgraph generated by $\hat{V}_s$.

Let $G = (V, E)$ be a graph and $U \subseteq V$. We denote the subgraph of G generated by $V \setminus U$ as $G \backslash U$.

Let $G = (V, E)$ be a graph and $H = (U, F)$ a subgraph of G, $G \backslash H$ is the induced subgraph of G generated by $V \setminus U$.

An online algorithm is a computable function $f : G_s \to Q_s$, where Q_s is some online structure such as a colouring. Thus, to list all such algorithms we would enumerate all partial computable functions $\{\phi_e \mid e \in \mathbb{N}\}$ of the correct kind. In the counterexamples below, such as Theorem 6.10, we will be diagonalizing against such lists. In that particular theorem, no algorithm will be able to online $2k-1$ colour the online structure. This method also gives asymptotic lower bounds for finite structures, so that if if G is sufficiently large, then it will defeat algorithms $1, \ldots, e$. Usually, we will have a "component" of the graph (or partial ordering) which is devoted to diagonalizing e. Of course, we will have achieved this diagonalization if the partial computable function does not halt. How this occurs is specific to the particular theorem.

6.1. **Colouring Strongly Online Graphs.**

A strategy introduced by Bean [10] is to colour odd stages with one set of colours and even stages with another set.

Theorem 1.6 was first proved by Bean [10] in the context of highly computable graphs. We present here a modification to the context of strongly online graphs.

Theorem 6.8. *If G is a strongly online graph with chromatic number $\chi(G)$ then G can be online coloured with $2\chi(G)$ many colours.*

Proof. Let $G = \lim_s G_s$ be a k-colourable strongly online graph.

We define the colouring at odd and even stages. In the even stages we use colours $1, \ldots, k$ and in the odd stages $k + 1, \ldots, 2k$.

We colour $G_0 = \emptyset$ with no colours.

For the even stage $2s+2$: Suppose we have coloured G_{2s+1} using $2k$ colours. Let v be the next vertex presented. Note that $G_{2s+2} = N[G_{2s+1} \cup \{v\}]$. At stage $2s+2$ we can see all of G_{2s+2}. Let $\bar{G}_{2s+2} = G_{2s+2} \backslash G_s$. We assign every vertex in $\bar{G}_{2s+2}$ a colour from $1, \ldots, k$. This can be done using brute force as G is k-colourable and $\bar{G}_{2s+2}$ is finite.

The odd stage is analogous, but we colour $\bar{G}_{2s+1}$ with colours from $k+1, \ldots, 2k$

This gives a colouring of G as $\bar{G}_s$ is disconnected from $\bar{G}_{s+2}$. □

In the context of highly computable graphs we have the following theorem.

Theorem 6.9 (Schmerl [64])**.** *If G is a highly computable, k-colourable graph then, G is computably $(2k - 1)$-colourable.*

Which would lead us to expect that we can improve on the bound in theorem 1.6. However this is not the case, theorem 1.6 is tight.

Theorem 6.10. *For all k there is a strongly online k-colourable graph that cannot be strongly $(2k - 1)$-online coloured.*

Proof. We construct an online presentation of graph as a series of disjoint subgraphs H which are presented themselves in an online fashion, so that $G = \bigcup_s H_s$.

Let $\langle \phi_e \rangle$ be an enumeration of all possible partial computable online colourings (i.e. functions $\mathbb{N} \mapsto [2k - 1]$). We diagonalize against ϕ_s by constructing a strongly online graph H_s that cannot be strongly online coloured by ϕ_s. We then let $G = \bigcup_s H_s$.

For component H_s, we will have constructed a strongly online graph is a sequence $H_{s,1}, H_{s,2}, \ldots$. Begin by letting $H_{s,1}$ be a k-clique. Let $H_{s,t+1}$ be $H_{s,t}$ together with with another k-clique and the neighbours of each k-path in $H_{s,t}$. (Thatis, we replicate $H_{s,t}$ and glue it on to $H_{s,t}$. See fig. 2. We repeat this until there are

$$n = 2(k - 1)\binom{2k - 1}{k - 1} + 1$$

k-paths. Let this graph be $H_{s,q}$. Let X_i be the last vertices revealed in each k-path in $H_{s,q}$. If ϕ_s is a valid colouring of $H_{s,t}$ then, it must halt on $H_{s,j}$ in order. Should

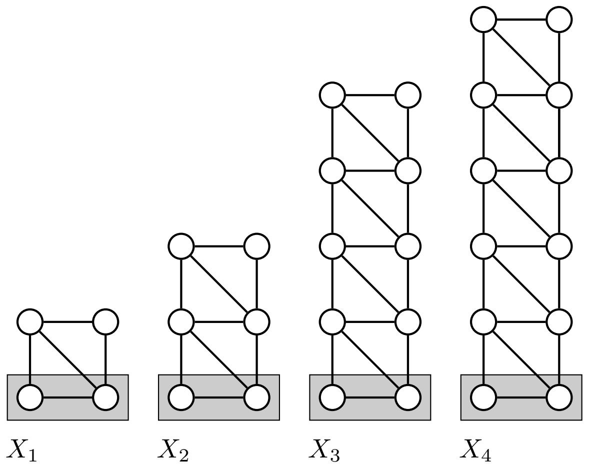

FIGURE 2. The first 4 paths for $H_{s,4}$ when $k = 3$

this occur, by the pigeonhole principle, ϕ_s must have coloured $2k$ X_i's using the same set of colours. Let I denote the set of indices of these sets.

Let $A = \big(\{a_1, \ldots, a_k\}, E\big)$ be a k-clique. We let $H_{s,q+1}$ be $H_{s,q} \cup A$ along with edges $\bigcup_{i \in I}\{(u, a_i) : u \in X_i\}$. See fig. 3. Without loss of generality we can assume that ϕ_s uses colours $1, \ldots, k-2$ to colour X_i. Then A will be coloured using $k-1, \ldots, 2k-1$.

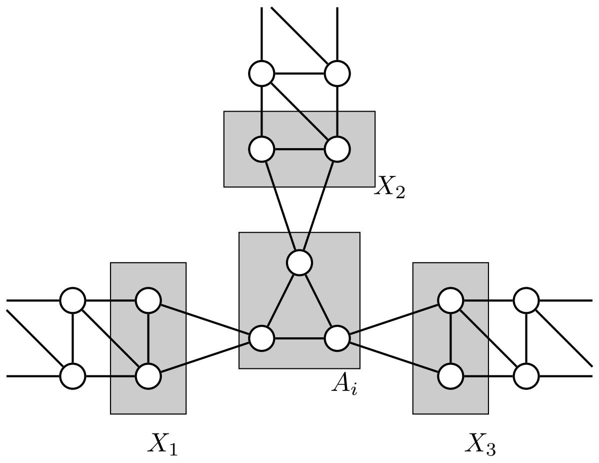

FIGURE 3. The graph $H_{s,q}$ for $k = 3$

We add $2k$ such cliques, each one connected to different X_i's. Let $\mathcal{A} = \{A_i : i \in J\}$ be the set all of these k-coloured cliques. Denote $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,k}\}$ where, for $j \in [1, k]$, $\phi_s\left(\bigcup_{i \in J}\{a_{i,j}\}\right)$ is a single colour. Let $B = (\{b_1, \ldots, b_k\}, E)$

be another k-clique. We let $H_{s,q+2}$ be $H_{s,q+1} \cup B$ along with the edges

$$\bigcup_{i \in J} \{(b_i, a_{2i-1,1}), \ldots, (b_i, a_{2i-1,k-1}), (b_i, a_{2i,k}), \ldots, (b_i, a_{2i,2k})\}$$

See fig. 4.

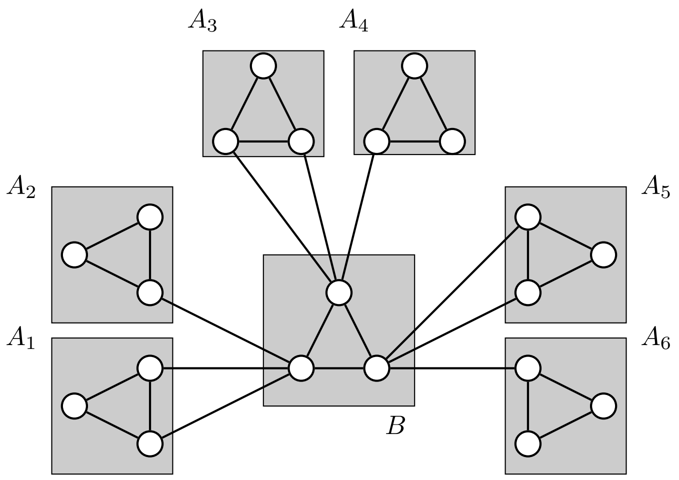

FIGURE 4. The graph H_s for $k = 3$, with colours

This ends the construction. Each vertex in B is connected to k uniquely coloured neighbours and is part of a k-clique, and thus cannot be $2k-1$-coloured. Therefore ϕ_s cannot strongly online colour H_s. A k-colouring of H_s can be found by colouring B then using breadth first search and the greedy strategy. Hence each component is k-colourable, and the components are disjoint.

Therefore, $G = \bigcup_s H_s$ is a k-colourable graph that cannot be coloured by any strategy. □

In the case of *locally* strongly online colouring trees, we cannot do better than online colouring.

Theorem 6.11. *For every online algorithm A and $n \in \mathbb{N}$ there is a locally strongly online graph G that cannot be online coloured by A. Indeed, for forests of height n, the approximation ratio is $\Omega(\log n)$ which is no better than the one for normal online trees.*

Proof. Fix $k \in \mathbb{N}$ and A an online algorithm. By induction on the number of colours $l \leq k$ we construct a locally strongly online forest T such that A does not give a locally strongly online k-colouring of T. We ensure that at each stage l the vertices that is needs the l-th colour are ones that have not been presented yet (but will have been revealed).

For $l = 0$ any single vertex is a tree and cannot be coloured with 0 colours. Let G be a pair of vertices a and b connected by an edge, and present the vertex a. b is revealed and requires 1 colour as desired.

Generate $2^k \cdot k^{k+1}$ many G's. By the pigeon hole principle $2^k \cdot k^k$ of the b vertices from these trees are the same colour. Let F_0 the set of such vertices.

Assume that for all $m \leq l$, $|F_m| \geq 2^{k-l} \cdot k^{k-l}$ and all vertices in F_m use the same colour, have not been presented, and are adjacent to $m-1$ different coloured neighbours. That is F_m uses the m-th colour.

Let $X = x_1, \ldots, x_n$ contain $2^{k-l-1} \cdot k^{k-l}$ many new vertices. Connected each x_i to a unique vertex from each $F_0, \ldots, F_l$ by presenting one of x_i's new neighbours. Each x_i is adjacent to l colours and so must get a new colour.

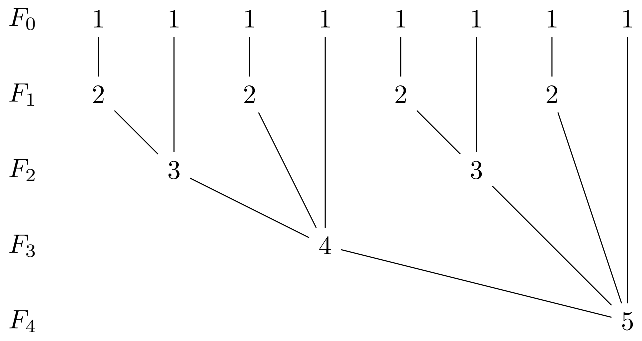

Figure 5. A tree that cannot be four coloured (numbers representing colours, and vertices not in a F_i omitted)

By the pigeon hole principle $2^{k-l-1} \cdot k^{k-l-1}$ many x_i's must use the same colour. Let F_{l+1} be these same coloured vertices.

As $2^{k-l-1} \cdot k^{k-l}$ is less than $1/2(2^{k-l} \cdot k^{k-l})$, each F_i decreases in size by no more than half. That is each F_i has $|F_i| \geq 2^{k-l-1} \cdot k^{k-l-1}$, as desired.

Note that once $l = k$ we will generate $2^{k-k} \cdot k^{k-k} = 1$ tree that requires a $k+1$-th colour. Therefore we have a forest that A cannot k-colour.

Finally note that, T has $O(2^k \cdot k^k)$ many vertices, and hence T requires $\Omega(\log(n))$ many colours to colour. □

Corollary 6.12 then follows as a simple corollary of theorem 6.3 by a diagonalization against all possible algorithms and numbers of colours.

Corollary 6.12. *There is an (infinite) locally strongly online forest which cannot be online finitely coloured.*

Proof. By theorem 6.3 for all $k \in \mathbb{N}$ and online algorithm A there is a forest $T_{k,A}$ that cannot be coloured by k colours. By a diagonalization of the forest $T = \bigcup_{k,A} T_{k,A}$ we have a forest that cannot be finitely coloured.

□

It is worth pointing out that if we use definition 3.2, then the functions ϕ_e in corollary 6.12 would all need to be primitive recursive and hence total. Then we would then be able to construct a tree that defeats all ϕ_e. This could be done by presenting vertices that join up two trees in the forest once they have been coloured.

However, we can do much better if we require that the strongly online graph is *strongly connected.* That is, at each stage s, T_s is presented as a tree in its online presentation.

Definition 6.13 (Strongly Connected)**.** A strongly online graph $G = \lim_s G_s$ with presentation $\langle G_s, H_s \rangle$ is *strongly connected* if at each stage H_s is connected.

Theorem 6.14. *Every strongly connected online tree is strongly 2-colourable.*

Proof. Let T be a strongly connected online tree. We colour T by induction on s, the stage.

For the base case we colour v_1 1 and $N(v)$ 2. This colours T_1.

Suppose we have coloured T_s. We must colour T_{s+1} Let $\hat{T}_{s+1} = T_{s+1} \backslash T_s$. Colour $\hat{T}_{s+1}$ with what ever colour we didn't use in $\hat{T}_s$.

As each v_s is part of the graph we can see, it has already been assigned a colour. And, as each $\hat{T}_s$ is independent, we have strongly 2-coloured T. □

6.2. **Graphs of Bounded Pathwidth.** As we have seen, many theorems for online structures seem to hold because of topological considerations/parameterizations about the shape of the graph. Graphs of bounded pathwidth are thought of as "path-like" and can have some nice online algorithms. Witness Kierstead's theorem ([54]) that any online graph of pathwidth k can be online $3k-2$-coloured. In the strongly online case we can do better. Now we know that if we are given a graph of pathwidth k as a pathwidth k presentation, that is bags of size at size at most $k+1$, then it can be $k+1$-coloured. Hence we know that in the strongly online case we can always $2k+2$ colour it. However, with a bit more work we can achieve the following bound which is tight by theorem 6.5.

Theorem 6.15. *Every strongly online graph with even pathwidth k is strongly online $2k+1$-colourable.*

Proof. Let $G = \lim_s G_s$ be a strongly online graph with pathwidth k.

Let

$$C_{s,t} = \begin{cases} 1, 3, 5, 7, \ldots, 2t-1 & \text{if } s \text{ is odd} \\ 2, 4, 6, 8, \ldots, 2t & \text{if } s \text{ is even} \end{cases}$$

At each stage s we will colour the boundary, $\hat{G}_s = (\hat{V}_s, \hat{E}_s)$, with the set of colours $C_{s,l}$ along with a $k+1$-th colour (which we will call *white* for convenience), where l is the pathwidth of $\hat{G}_s$.

When colouring $\hat{G}_s$ we will ensure that the colour white is only used in a connected component in $\hat{G}_s$ that has pathwidth k. All other components will use the colours $C_{s,m+1}$, where m is the pathwidth of the connected component.

The first stage ($s = 1$) has pathwidth at most k and so can be coloured appropriately with $C_{1,k}$ plus white.

Suppose we have coloured G_s, and we need to colour $\hat{G}_{s+1}$. Let H be a connected component in $\hat{G}_{s+1}$. Suppose that H has pathwidth $m < k$, then as the only neighbours of H have colours $C_{s,k}$ plus white, we can colour H with $C_{s+1,m+1}$.

Now, suppose that H has pathwidth k. We must now find a way to reuse the colour white. Let $\mathcal{P} = \{P_1, P_2, P_3, \ldots\}$ be a path decomposition of $H \cup N(H)$. There are three cases.

(1) H is adjacent to no components of width k in $\hat{G}_s$.
(2) H is adjacent to at most 1 component of width k in $\hat{G}_s$.
(3) H is adjacent to more than 1 component of width k in $\hat{G}_s$.

Case 1: H is not adjacent to any white vertices. Thus any colouring of H using $C_{s+1,k}$ plus white, is valid.

Case 2: Let P_i be a bag in $\mathcal{P}$ that contains a white vertex from $\hat{G}_s$. Without loss of generality assume that i is less than the index of any bag that contains only vertices from H. That is P_i is to the left of H. We use Algorithm 1 to find $X \subseteq V(H)$ such that X is 1-colourable, $H \backslash X$ has pathwidth $k-1$, and no vertex in X is adjacent to a white vertex in G_s.

Algorithm 1 Find the vertices of H to colour white

```
1: procedure FindVertices($i$)
2:     $j \leftarrow i$
3:     $X \leftarrow \emptyset$
4:     while $P_j \neq \emptyset$ do
5:         if $P_J \subseteq V(H)$ and $P_j \cap X = \emptyset$ then
6:             Fix some $x \in P_j \setminus P_{j-1}$
7:             $X \leftarrow X \cup \{x\}$
8:         end if
9:         $j \leftarrow j+1$
10:    end while
11:    return $X$
12: end procedure
```

Colour all vertices in X, and the remaining vertices in H with $C_{s+1,k}$.

Case 3: Assume for a contradiction that H is connected to 3 components, D,E, and F from $\hat{G}_s$ of width K. Because each of D, E, F,and H have pathwidth K there are bags P_d, P_e, P_f, and P_h such that $P_d \subset V(D)$, $P_e \subset V(E)$,$P_f \subset V(F)$, $P_h \subset V(H)$, and $|P_d| = |P_e| = |P_f| = |P_h| = k+1$.

Without loss of generality assume that $d < h < e < f$. As there is a path from P_h to D_f every bag between P_h and D_f must contain a vertex from $H \cup F$. This contradicts the fact that $h < e < f$. Therefore H is connected to two components of pathwidth k.

Claim 6.15.1. *Every Bag $P_i \in \mathcal{P}$ such that $P_i \subseteq V(H)$ and $|P_i| = k+1$ at most ${}^{k+1}/_2$ vertices are adjacent to k+1 colours from $\{P_1, \dots, P_{i-1}\}$ (or $\{P_{i+1}, P_{i+2}, \dots\}$).*

Proof of claim. Fix $P_i \in \mathcal{P}$ such that $P_i \subseteq V(H)$ and $|P_i| = k+1$. Let $\{p_1, \dots, p_n\}$ be the vertices in P_i adjacent to $k+1$ colours from $\{P_1, \dots, P_{i-1}\}$.

Let P_j be the greatest $j < i$ such that P_j contains $k+1$ colours (cannot be in H as H has not been coloured). No bag between P_j and P_i can contain white (or else there would be a third component of with k). Let y be the white vertex in P_j. $p_1, \dots, p_n$ are adjacent to y, further there is some bag P_l such that $p_1, \dots, p_n, y \in P_l$.

Note that the maximum width of a component from $\hat{G}_s$ between P_l and P_i is $k-n$. Hence $p_1, \dots, p_n$ can only be adjacent to less than $k+1-n$ colours not in P_l.

In P_{l-1} there are only $n-1$ vertices from $p_1, \dots, p_n$. Hence the maximum number of colours from P_j that $p_1, \dots, p_n$ is adjacent to is $k+1-n$.

If $n > {}^{k+1}/_2$ then the number of colours not from P_l and from P_j is

$$(k+1-n) + (k+1-n) < 2k+2-2({}^{k+1}/_2) = k+1$$

Which contradicts the fact that $p_1, \dots, p_n$ are adjacent to $k+1$ colours from $\{P_1, \dots, P_{i-1}\}$. Therefore n is at most ${}^{k+1}/_2$. ■

Let A and B be the components of pathwidth k that H is adjacent to.

Let P_l be the leftmost and P_r the rightmost bag in $\mathcal{P}$ such that $P_l \subseteq V(H)$ and $P_r \subseteq V(H)$. By claim 6.15.1 and the fact that k is even, P_l and P_r have more than ${}^{k+1}\!/_2$ many vertices adjacent to at most k colours from $\hat{G}_s$. [12]

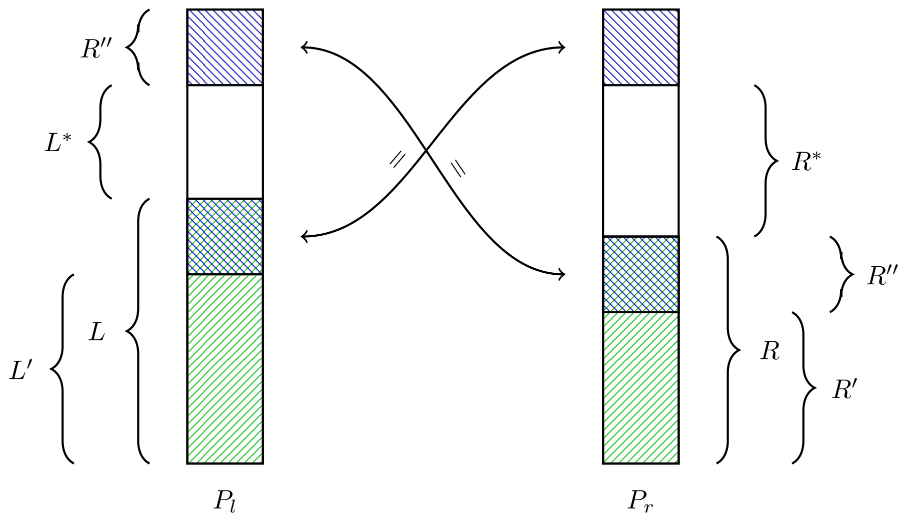

Figure 6. The bags P_l and P_r

We define the following subsets of P_l and P_r.

- $L \subseteq P_l$ be the subset of P_l that is not connected to $k+1$ colours from $\{P_1, \dots, P_{l-1}\}$.
- $R \subseteq P_r$ be the subset of P_r that is not connected to $k+1$ colours from $\{P_{r+1}, P_{r+2}, \dots\}$.
- $L' \subseteq L$ such that for every $x \in L'$, $x \notin P_r$.
- $R' \subseteq R$ such that for every $y \in R'$, $y \notin P_l$.
- $L^* = P_l \setminus L \setminus P_r$.
- $R^* = P_r \setminus R \setminus P_l$.
- $R'' = R \setminus R'$ ($= P_l \setminus L \setminus L^*$).

Note that $|L^*| = k+1-|L|-|R''|$ and $|R'| = |R|-|R''|$. Hence

$$|L^*| \leq k+1-|{}^{k+1}\!/_2|-|M| < {}^{k+1}\!/_2-|M| \leq |R'|$$

Thus $|L^*| < |R'|$.

Therefore any colouring of H must have one colour that is used in both L' and R'. Let $x \in L'$ and $y \in R'$ be some such vertices. If both x and y are not adjacent to the colour white, then we can colour these vertices white and are done.

Suppose one of x and y is adjacent to the colour white. Then this means one of L^* and R^* (respectively) has one less vertex in it. Hence L' (or R') is one vertex larger. Therefore there in any colouring of H there are two colours used in both L' and R'.

[12]Note that this fact that fails to be true if k is odd. We could have exactly half of the vertices in H connected to $k+1$ colours to the left and the other half connected to $k+1$ colours to the right. Thus we might not be able to colour H.

By applying induction we see that either we will be able to colour H or one of $L = P_l$ or $R = R_r$. We then use the technique from case 2 to colour H as A or B effectively has pathwidth less than k.

It remains to show that no neighbour of H is later stages can have pathwidth k.

Claim. *If F is a connected component in $G \setminus G_s$ that is connected to H, then F has pathwidth less than k.*

Proof of claim. Let $\mathbb{Q} = \{Q_1, Q_2, \dots\}$ be a path decomposition of G. Because each of A, B, and H have pathwidth k there must be bags Q_a, Q_b, and Q_h such that $Q_a \subset V(A)$, $Q_b \subset V(B)$, $Q_h \subset V(H)$, and $|Q_a| = |Q_b| = |Q_h| = k + 1$.

Assume for a contradiction that F has pathwidth k. Then there is also a bag $Q_f \subseteq V(F)$, where $|Q_f| = k + 1$.

In G there are paths Q_a to Q_h and Q_b to Q_h. $Q_f \cap V(H) = Q_f \cap V(A) = Q_f \cap V(B) = \emptyset$ and so, f cannot be between a and b because the paths cannot pass through Q_f. That is $a > f$ and $b < f$.

f cannot be greater than b because there is a path from Q_f to Q_h and $Q_b \cap (V(h) \cup V(F)) = \emptyset$. Similarly f cannot be less than a.

This is a contradiction as Q_f must exist somewhere in $\mathbb{Q}$. Therefore F cannot have pathwidth k. ■

This means that we can always use k colours to colour later neighbours of H. □

The following result shows that we cannot do better than $2k + 1$.

Theorem 6.16. *There is a strongly online graph of pathwidth k that is not strongly online $2k$-colourable.*

Proof. As in theorem 6.10 we construct the graph in stages. Let $\langle \phi_e \rangle$ be an enumeration of all possible colourings (i.e. functions $\mathbb{N} \mapsto [2k]$). Again the graph is constructed as disjoint components and we will diagonalize against ϕ_s by constructing a strongly online graph H_s with path width k that cannot be strongly online coloured by ϕ_s. We then let $G = \bigcup_s H_s$. We return to the construction of component H_s which is constructed in stages e.

Stage e: Recall that a strongly online graph is a sequence $H_{s,1}, H_{s,2}, \dots H_{s,e-1} \dots$. We construct a series of k-paths, each with vertices $x_1, x_2, x_3, \dots$ and all the edges of the form (x_i, x_j) such that $i - j < k$. Let $\{P_1, P_2, P_3, \dots\}$ denote the set of all such k-paths. Let $H_{s,1}$ contain P_1 with k vertices. Let $H_{s,i+1}$ contain P_{i+1} with k vertices and all P_j such that $j < i$ with an extra k vertices. See fig. 2 for an example when $k = 3$.

We repeat this until there are

$$n = \binom{2k}{k-1} + 1$$

k-paths. Now if it is the case that ϕ_s halts we will observe the online colouring it has given. By the pigeonhole principle there are two paths whose last $k-1$ vertices have the same colour. Denote these vertex sets $A = \{a_1, \dots, a_k\}$ and $B = \{b_1, \dots, b_k\}$.

Without loss of generality assume that $c(a_i) = c(b_i) = i$. Let D be a $(k+1)$-clique with vertex set $\{d_1, \dots, d_{k+1}\}$. We let $H_{s,n+1} = H_{s,n} \cup D$ along the edges of the form (d_i, a_j) such that $i \leq j$ and (d_i, b_j) such that $i > j$. This forms a single k-path See fig. 7.

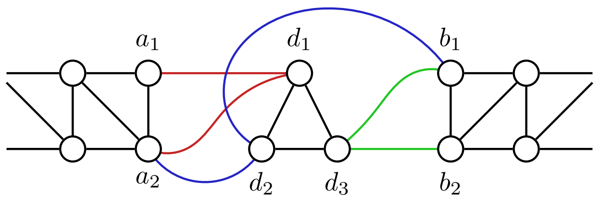

FIGURE 7. The graph $H_{s,n+1}$ with edges coloured for clarity

Each vertex in D is connected to a vertex of each colour $1, \dots, k-1$ and thus must be coloured from the colours $k, \dots, 2k$. However as D is a $(k+1)$-clique it requires $k+1$ colours. Hence D cannot be coloured. Therefore H_s cannot be coloured by ϕ_s.

H_s is a series of k-paths and so has pathwidth k.

Therefore, $G = \bigcup_s H_s$ is a graph with pathwidth k that cannot be coloured by any strategy. □

For graphs with odd pathwidth we cannot do any better than than theorem 1.6 as evidenced by theorem 6.6.

Theorem 6.17. *There is a strongly online graph of odd pathwidth k that is not strongly online $2k+1$colourable.*

As we have done previously the proof of theorem 6.6 proceeds by a diagonalization against a list of all partial computable colourings. For each such colouring we generate two $k+1$ cliques (X_1 and X_2) and two $k+1/2 = l$ cliques (Y_1 and Y_2) such that each pair uses the same colours. We then reveal another clique of size $k+1$ that is connected to $k+1$ colours. Half the colours coming from X_1 and X_2 and the other half coming from Y_1 and Y_2. This new clique will not be colourable with the remaining k colours.

Proof. As in as theorem 6.10 we construct the graph in stages. Let $\langle \phi_e \rangle$ be an enumeration of all possible colourings (i.e. functions $\mathbb{N} \mapsto [2k]$). The graph is constructed as disjoint components and we will diagonalize against ϕ_s by constructing a strongly online graph H_s with path width k that cannot be strongly online coloured by ϕ_s in $2k+1$ colours.

Note that because k is odd $k+1 = 2l$ for some $l \in \mathbb{N}$.

Stage e: To construct H_s we begin by presenting

$$2\binom{2k+1}{k+1} + \binom{2k+1}{l} + 2$$

vertices each the center of a path that grows in length with every presented vertex.

Next we create $\binom{2k+1}{k+1} + 1$ cliques of size $k+1$ and $\binom{2k+1}{l} + 1$ cliques of size l.

If it is the case that ϕ_e halts we will observe the following. First, there will be two cliques X_1 and X_2 of size $k+1$ which use the same colours. Second, there are two cliques Y_1 and Y_2 of size l which use the same colours. Without loss of generality let the colours used in Y_1 and Y_2 be $1, 2, \dots, l$, and the colours used in X_1 and X_2 be $1, 2, \dots, k+1$.

We join the vertices coloured $1, \dots, l$ in X_1 (respectively X_2) to the end of one of our paths, call this vertex a_1 (respectively a_2). We do the same for the vertices coloured $l+1,\dots,k+1$ in X_1 (respectively X_2), who join to the end of one of our paths, call this vertex b_1 (respectively b_2).

Next join half of Y_1 (respectively X_2) to some path, call this c_1 (respectively c_2), and the other half to some path and call it d_1 (respectively d_2).

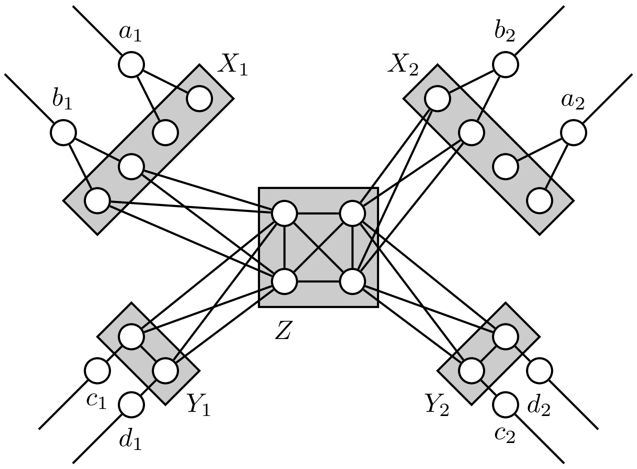

FIGURE 8. The graph H_s for pathwidth 3.

Next we reveal another clique, Z, of size $k+1$ with vertices $z_1,\dots,z_{k+1}$. For each $i \leq l$ join the vertex z_i to every vertex in Y_1 and every vertex with colour greater than l in X_1. For $i > l$ connect z_i to every vertex in Y_2 and every vertex with colour greater than l in X_2 See fig. 8, for an example.

Each vertex in Z is connected to $k+1$ colours (colours $1,\dots,l$ from Y_i and colours $l+1,\dots,k+1$ from X_i) and is part of a $k+1$ sized clique. Therefore Z cannot be coloured with k colours.

It remains to show that H_s has pathwidth k. The path decomposition of H_s contains the following bags in order,

(1) $a_1 \cup N(a_1) \cap X_1$
(2) X_1
(3) $b_1 \cup N(b_1) \cap X_1$
(4) $(N(X_1) \cap Z) \cup (N(Z) \cap X_1)$
(5) Path adjacent to c_1
(6) $c_1 \cup N(c_1) \cap Y_1$
(7) Y_1
(8) $d_1 \cup N(d_1) \cup Y_1$
(9) Path adjacent to d_1
(10) Z
(11) Path adjacent to d_2
(12) $d_2 \cup N(d_2) \cup Y_2$
(13) Y_2
(14) $c_2 \cup N(c_2) \cup Y_2$
(15) Path adjacent to c_2
(16) $(N(X_2) \cap Z) \cup (N(Z) \cap X_2)$

(17) $b_2 \cup N(b_2) \cap X_2$
(18) X_2
(19) $a_2 \cup N(a_2) \cap X_2$

Along with the interpolation of any necessary vertices. By the construction of H_s each of these bags will contain no more than $k+1$ vertices. Hence H_s has pathwidth k.

Therefore $G = \bigcup_s H_s$ is a strongly online graph with odd pathwidth k that cannot be $2k+1$-coloured. □

6.3. **Strongly Online Pathwidth.** In view of the usefulness of graph width metrics in classical graph theory, we would hope that there should be online versions of the same. What is the best we can do for an online graph where we are required to construct some kind of decomposition online. We offer the following possibility. We remark that originally we hoped that if we could construct an online path decomposition of width $2k$ for a (strongly) online graph of width k, then we could use a simple greedy algorithm to online colour it with $2k+1$ many colours. Alas this is not the case as we soon see.

Definition 6.18 (Path Decomposition Extension)**.** Let $\mathcal{P}$ and $\mathbb{Q}$ be path decompositions. $\mathbb{Q}$ is an *extension* of $\mathcal{P}$ (denoted $\mathcal{P} \preceq \mathbb{Q}$) if there is an injection $\sigma\colon \mathcal{P} \to \mathbb{Q}$ such that for all $P_i \in \mathcal{P}, P_i \subseteq \sigma(P_i)$. For each P_i, P_j such that $P_i \cap P_j \neq \emptyset$, if $i < j$, then $\sigma(P_i) = Q_l$ and $\sigma(P_j) = Q_m$ for some $l < m$.

Definition 6.19 (Strongly Online Path Decomposition)**.** Let $G = \lim_s G_s$ be a strongly online graph. A *strongly online path decomposition* is a sequence $\langle \mathcal{P}_s \rangle_s$ such that

(1) Each $\mathcal{P}_s$ is a path decomposition of $G_s^{\prec}$
(2) $\mathcal{P}_s$ is an extension of $\mathcal{P}_{s-1}$
(3) $\mathcal{P} = \lim_s \mathcal{P}_s$ is a path decomposition of G

That is, using the information in G_s the vertex v_s must be place inside at least one bag. Other vertices may be placed in bags (including vertices already in a bag), but vertices cannot be removed.

Definition 6.20. The *strongly online pathwidth* of a graph G is the smallest k such that for each strongly online presentation $G = \lim_s G_s$, G has a strongly online path decomposition of width at most k.

Theorem 6.21. (1) *For each online algorithm A and finite number n, there is a strongly online (finite) G of pathwidth 2 which cannot have a online path decomposition of width n. We can construct G with at most $O(n^4)$ many vertices.*
(2) *There is an (infinite) strongly online graph of pathwidth 2 which cannot have a online path decomposition*

The basic idea is for each possible online algorithm A that can rearrange the bags from stage to stage present $n(2n+2)$ vertices that each form part of a single path. Then use A to determine the order in which the bags containing each of the $n(2n+2)$ vertices will be ordered after we reveal all the vertices that join them. We join the paths (bags) across the middle bag $2n$ times. This forces the middle bag to have at least $n+1$ vertices in it, as A has made the wrong choices.

Again we will be diagonalizing against partial computable functions ϕ_e and we will do so in pairs $\langle e, n\rangle$ where ϕ_e is claiming to give an online path decomposition of width n. For this we will again use disjoint components for the graph. These might keep growing forever, if ϕ_e fails to respond.

Proof. Fix $n \in \mathbb{N}$, and A an online algorithm. Note that at each stage A gives us a path decomposition of G_s. Because we want to force a pathwidth greater than $n-1$ every bag has size at most n, we will force at least 1 bag to have size $n+1$. By using a pairing function we consider A as a way of extending a path decomposition as in definition 6.18. We preform a computable construction.

We construct a strongly online graph $G = \lim_t G_t$.

We build G in two stages. In stage 1, we present $n(2n+1)$ vertices each the center of a path that grows in length with every presented vertex. Once this is done let $\mathcal{P}$ be path decomposition generated by A on the currently presented vertices, this signals the start of stage 2.

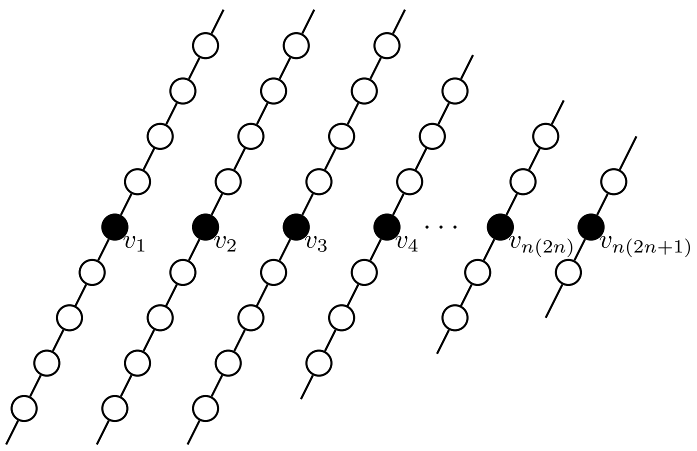

FIGURE 9. The initial $n(2n+1)$ paths and the presented vertices (in black)

In stage 2 there are two cases. First, if $A(\mathcal{P})$ does not give us a valid decomposition or a valid extension of $\mathcal{P}$ then G satisfies our requirements and we stop. Second, $A(\mathcal{P})$ is a valid decomposition. In which case, because $A(\mathcal{P})$ has $n(2n+1)$ vertices divided into bags of size n there must be $2n+1$ bags each containing a vertex not in any of the other $2n+1$ bags. Let these bags be $X_1, \ldots, X_{2n+1}$, and the corresponding vertices $v_1, \ldots, v_{2n+1}$. Without loss of generality assume that $X_1, \ldots, X_{2n+1}$ are ordered by their appearance in $A(\mathcal{P})$.

Each X_i corresponds to a path in G. We reveal $2n$ vertices each one joining the end of two paths in G_e as follows. For $i \leq n$, join the i-th path to the $(2n+2-i)$-th and the $(2n+1-i)$-th. This forms a sort of spiral (see fig. 10).

We now stop revealing vertices in G and present all the vertices visible in the order they were revealed. This ends the construction of G.

We need to show that A does not give a valid path decomposition of G. Fix $i \leq n$. Let $u_1, u_2, \ldots, u_a$ be the path $v_i - v_{2n+2-i}$, $\mathbb{Q}$ be the current path decomposition

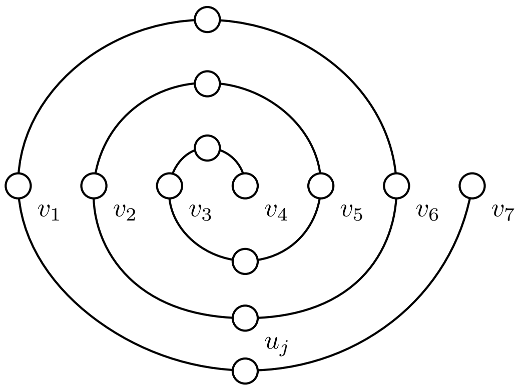

FIGURE 10. The connections between vertices (paths) for $n = 3$

given by A and $U_1, U_2, \ldots, U_b$ be the bags in $\mathbb{Q}$ that contain a u_j, in the order they appear in $\mathbb{Q}$. Note that $u_1 = v_i$ and $u_a = v_{2n+2-i}$. Without loss of generality we can assume that $U_1 = X_i$ and $U_b = U_{2n+2-i}$ (see fig. 11).

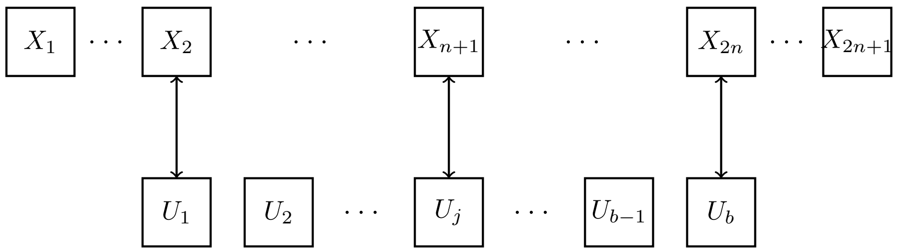

FIGURE 11. The bags in $\mathbb{Q}$ and the bags corresponding to the path $v_i - v_{2n+2-i}$.

Thus X_{n+1} appears between U_1 and U_b. Hence there must be some U_j such that $U_j = X_{n+1}$. Therefore X_{n+1} contains some u_j. Then by letting i vary we can see that X_{n+1} must contain $n + 1$ vertices (one from each path along with the vertex v_{n+1}). Therefore $A(\mathcal{P})$ does not give a width $n - 1$ path decomposition of G.

For each i the path $v_i - v_{2n+1-i}$ forces a vertex into X_{n+1} just as above. However this may be the same vertex forced in by $v_i - v_{2n+2-i}$. Thus from the $2n$ paths only half are guaranteed to increase the size of X_{n+1}.

G is union of $n(2n + 1)$ paths. Because at each stage each path gets another two vertices the i-th path has $1 + 2(i - 1)$ vertices. Finally we add another $2n$ vertices to join these paths together. Therefore the total number of vertices is

$$\sum_{i=1}^{2n^2+n} \big(1 + 2(i - 1)\big) + 2n = n^2(1 + 2n)^2 + 2n = O(n^4)$$

Then by a diagonalization against all possible strategies and widths we can see that we have a strongly online graph with no path strongly online path decomposition. □

References

[1] P. E. Alaev, *Structures Computable in Polynomial Time. I*, Algebra and Logic **55** (2017), no. 6, 421–435.

[2] ———, *Categoricity for Primitive Recursive and Polynomial Boolean Algebras*, Algebra and Logic **57** (2018), no. 4, 251–274.

[3] ———, *Structures Computable in Polynomial Time. II*, Algebra and Logic **56** (2018), no. 6, 429–44.

[4] Susanne Albers, *Online algorithms: a survey*, Mathematical Programming **97** (2003), no. 1, 3–26.

[5] Noga Alon, Raphael Yuster, and Uri Zwick, *Color-coding*, J. Assoc. Comput. Mach. **42** (1995), no. 4, 844–856.

[6] Jeremy Avigad, *Inverting the furstenberg correspondence*, Discrete and Continuous Dynamical Systems **32** (2012), no. 10.

[7] Nikolay Bazhenov, Rod Downey, Iskander Kalimullin, and Alexander Melnikov, *Foundations of online structure theory*, jun 2019, pp. 141–181.

[8] Nikolay Bazhenov, Matthew Harrison-Trainor, Iskander Kalimullin, Alexander Melnikov, and Keng Meng Ng, *Automatic and polynomial-time algebraic structures*, Journal of Symbolic Logic **84** (2019), no. 4.

[9] Nikolay Bazhenov, Iskander Kalimullin, Alexander Melnikov, and Keng Meng Ng, *Online presentations of finitely generated structures*, Theoretical Computer Science **844** (2020).

[10] Dwight R Bean, *Effective Coloration*, The Journal of Symbolic Logic **41** (1976), no. 2, 469–480.

[11] K. V. Blinov, *Primitively Recursively Categorical Linear Orderings*, Siberian Mathematical Journal **60** (2019), no. 1, 20–26.

[12] Hans L. Bodlaender, *A linear time algorithm for finding tree-decompositions of small treewidth*, Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, STOC '93, Association for Computing Machinery, 1993, p. 226234.

[13] Hans L. Bodlaender and Ton Kloks, *Efficient and constructive algorithms for the pathwidth and treewidth of graphs*, J. Algorithms **21** (1996), no. 2, 358402.

[14] Boštjan Brešar, Sandi Klavžar, and Douglas F. Rall, *Domination game and an imagination strategy*, SIAM J. Discrete Math. **24** (2010), no. 3, 979–991.

[15] Csilla Bujtás, *Domination game on forests*, Discrete Math. **338** (2015), no. 12, 2220–2228.

[16] Douglas Cenzer and Jeffrey B. Remmel, *Complexity theoretic model theory and algebra*, Handbook of recursive mathematics, Vol. 1, of Stud. Logic Found. Math., vol. 138, North-Holland, Amsterdam, 1998, pp. 381–512.

[17] Chen Chung Chang and H. Jerome Keisler, *Continuous Model Theory. (AM-58)*, Princeton University Press, 1966.

[18] Marek Chrobak and Maciej Ślusarek, *On some packing problem related to dynamic storage allocation*, RAIRO Inform. Théor. Appl. **22** (1988), no. 4, 487–499.

[19] Bruno Courcelle, *The monadic second-order logic of graphs. I. Recognizable sets of finite graphs*, Information and Computation **85** (1990), no. 1, 12–75.

[20] Eamonn A. O'Brien Derek F. Holt, Bettina Eick, *Handbook of computational group theory*, Chapman and Hall/CRC, Boca Raton, Florida, 2005.

[21] John Doner and Wilfrid Hodges, *Alfred Tarski and decidable theories*, J. Symbolic Logic **53** (1988), no. 1, 20–35.

[22] Rod Downey, *Computability, definability and algebraic structures*, Proceedings of the 7th and 8th Asian Logic Conferences, Singapore Univ. Press, Singapore, 2003, pp. 63–102.

[23] Rod Downey, Alexander Melnikov, and Keng Meng Ng, *Foundations of online structure theory II: The operator approach*, Log. Methods Comput. Sci. **17** (2021), no. 3, Paper No. 6, 35.

[24] Rodney Downey and Michael Fellows, *Parameterized complexity*, Springer-Verlag, 1999.

[25] ———, *Fundamentals of parameterized complexity*, Springer-Verlag, 2013.

[26] Rodney Downey, Matthew Harrison-Trainor, Iskander Kalimullin, Alexander Melnikov, and Daniel Turetsky, *Graphs are not universal for online computability*, Journal of Computer and System Sciences **112** (2020), 1–12.

[27] Rodney Downey and Denis Hirschfeldt, *Algorithmic randomness and complexity*, Springer-Verlag, 2010.

[28] Rodney Downey and Catherine McCartin, *Online problems, pathwidth, and persistence*, International Workshop on Parameterized and Exact Computation, Lecture Notes in Computer Science, Springer-Verlag, 2004, pp. 13–24.
[29] Rodney Downey and Dimitrios Thilikos, *Confronting intractability via parameters*, Computer Science Review **5** (2011), no. 4, 279–317.
[30] Rodney G Downey and Catherine McCartin, *Online promise problems with online width metrics*, Journal of Computer and System Sciences **73** (2007), no. 1, 57–72.
[31] David B. A. Epstein, James W. Cannon, Derek F. Holt, Silvio V. F. Levy, Michael S. Paterson, and William P. Thurston, *Word processing in groups*, Jones and Bartlett Publishers, Boston, MA, 1992.
[32] A. Fröhlich and J. C. Shepherdson, *Effective procedures in field theory*, Philos. Trans. Roy. Soc. London Ser. A **248** (1956), 407–432.
[33] Haim Gaifman, *On local and non-local properties*, Studies in Logic and the Foundations of Mathematics, vol. 107, 1982, pp. 105–135.
[34] W. Gasarch, *A survey of recursive combinatorics*, Handbook of recursive mathematics, Vol. 2, Stud. Logic Found. Math., vol. 139, Elsevier, 1998, pp. 1041–1176.
[35] Sergei S. Goncharov, *Countable Boolean algebras and decidability*, Siberian School of Algebra and Logic, Consultants Bureau, New York, 1997.
[36] Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein, *Finite model theory and its applications*, Texts in Theoretical Computer Science. An EATCS Series, Springer, Berlin, 2007.
[37] Serge Grigorieff, *Every recursive linear ordering has a copy in* DTIME-SPACE$(n, \log(n))$, J. Symbolic Logic **55** (1990), no. 1, 260–276.
[38] A. Gyárfás and J. Lehel, *On-line and first fit colorings of graphs*, J. Graph Theory **12** (1988), no. 2, 217–227.
[39] ———, *On-line and first fit colorings of graphs*, J. Graph Theory **12** (1988), no. 2, 217–227.
[40] J. Hartmanis and R. E. Stearns, *On the computational complexity of algorithms*, Trans. Amer. Math. Soc. **117** (1965), 285–306.
[41] Carl G Jockusch Jr and Robert I Soare, *Degrees of orderings not isomorphic to recursive linear orderings*, Annals of pure and applied logic **52** (1991), no. 1-2, 39–64.
[42] Iskander Kalimullin, Alexander Melnikov, and Keng Meng Ng, *Algebraic structures computable without delay*, Theoret. Comput. Sci. **674** (2017), 73–98.
[43] ———, *The diversity of categoricity without delay*, Algebra and Logic **56** (2017), 171–177.
[44] Richard M. Karp, *Reducibility among combinatorial problems*, Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972) (R.E. Miller, J.W. Thatcher, and J.D. Bohlinger, eds.), The IBM Research Symposia Series, Plenum Press, New York, 1972, pp. 85–103.
[45] Richard M Karp, *On-line algorithms versus off-line algorithms: How much*, Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, vol. 1, North-Holland Publishing Co., 1992, pp. 416–429.
[46] Bakhadyr Khoussainov and Anil Nerode, *Automatic presentations of structures*, Logic and computational complexity (Indianapolis, IN, 1994), Lecture Notes in Comput. Sci., vol. 960, Springer, Berlin, 1995, pp. 367–392.
[47] H. A. Kierstead, *The linearity of first-fit coloring of interval graphs*, SIAM J. Discrete Math. **1** (1988), no. 4, 526–530.
[48] ———, *Recursive and on-line graph coloring*, Handbook of recursive mathematics, Vol. 2, Stud. Logic Found. Math., vol. 139, North-Holland, Amsterdam, 1998, pp. 1233–1269.
[49] H. A. Kierstead and Jun Qin, *Coloring interval graphs with First-Fit*, vol. 144, 1995, Combinatorics of ordered sets (Oberwolfach, 1991), pp. 47–57.
[50] H. A. Kierstead, David A. Smith, and W. T. Trotter, *First-fit coloring on interval graphs has performance ratio at least 5*, European J. Combin. **51** (2016), 236–254. MR 3398852
[51] Hal A Kierstead and William T Trotter, *Planar graph coloring with an uncooperative partner*, Journal of Graph Theory **18** (1994), no. 6, 569–584.
[52] Henry A. Kierstead, *Recursive and on-line graph coloring*, Handbook of recursive mathematics, Vol. 2, Stud. Logic Found. Math., vol. 139, North-Holland, Amsterdam, 1998, pp. 1233–1269.
[53] Henry. A. Kierstead, *A simple competitive graph coloring algorithm*, Journal of Combinatorial Theory, Series B **78** (2000), no. 1, 57–68.

[54] Henry A. Kierstead and William T. Trotter, Jr., *An extremal problem in recursive combinatorics*, Congr. Numer. **33** (1981), 143–153.

[55] William B. Kinnersley, Douglas B. West, and Reza Zamani, *Extremal problems for game domination number*, SIAM J. Discrete Math. **27** (2013), no. 4, 2090–2107.

[56] Leonid Libkin, *Elements of finite model theory*, Springer-Verlag, 2004.

[57] László Lovász, Michael Saks, and W. T. Trotter, *An on-line graph coloring algorithm with sublinear performance ratio*, vol. 75, 1989, Graph theory and combinatorics (Cambridge, 1988), pp. 319–325.

[58] Matthew Askes, *Online and Adverserial Online Games*, MSc thesis, Victoria University of Wellington, 2022.

[59] N. S. Narayanaswamy and R. Subhash Babu, *A note on first-fit coloring of interval graphs*, Order **25** (2008), no. 1, 49–53.

[60] Christos H. Papadimitriou and Mihalis Yannakakis, *On limited nondeterminism and the complexity of the V-C dimension*, vol. 53, 1996, Eighth Annual Structure in Complexity Theory Conference (San Diego, CA, 1993), pp. 161–170.

[61] Jeffrey B. Remmel, *Recursive Boolean algebras*, Handbook of Boolean algebras, Vol. 3, North-Holland, Amsterdam, 1989, pp. 1097–1165.

[62] Joseph G. Rosenstein, *Linear orderings*, Pure and Applied Mathematics, vol. 98, Academic Press, New York-London, 1982.

[63] Marek Cygan Fedor V. Fomin Lukasz Kowalik Daniel Lokshtanov D'/aniel Marx Marcin Pilipczuk Michal Pilipczuk Saket Saurabh, *Parameterized algorithms*, Springer, 2015.

[64] James H. Schmerl, *Recursive colorings of graphs*, Canadian J. Math. **32** (1980), no. 4, 821–830.

[65] Charles C. Sims, *Computation with finitely-presented groups*, vol. 48, Cambridge University Press, Cambridge, 1994.

[66] Daniel D. Sleator and Robert E. Tarjan, *Amortized efficiency of list update and paging rules*, Comm. ACM **28** (1985), no. 2, 202–208.

[67] Alfred Tarski, *Arithmetical classes and types of boolean algebras*, Bulletin of the American Mathematical Society **55** (1949).

[68] Todor Tsankov, *The additive group of the rationals does not have an automatic presentation*, The Journal of Symbolic Logic **76** (2011), no. 4, 1341–1351.

[69] A. M. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem*, Proc. London Math. Soc. (2) **42** (1936), no. 3, 230–265.

[70] H.S Witsenhausen, *On woodall's interval problem*, Journal of Combinatorial Theory, Series A **21** (1976), no. 2, 222–229.

[71] Jiaojiao Wu and Xuding Zhu, *Lower bounds for the game colouring number of partial k-trees and planar graphs*, Discrete Math. **308** (2008), no. 12, 2637–2642. MR 2410475

[72] Itaï Ben Yaacov, Alexander Berenstein, C. Ward Henson, and Alexander Usvyatsov, *Model theory for metric structures*, Model Theory with Applications to Algebra and Analysis, vol. 2, Cambridge University Press, 2010, pp. 315–427.

[73] Xuding Zhu, *Refined activation strategy for the marking game*, J. Combin. Theory Ser. B **98** (2008), no. 1, 1–18. MR 2368019

Victoria University of Wellington, `matthewaskes@gmail.com, Rod.Downey@msor.vuw.ac.nz`