Deterministic Local Algorithms, Unique Identifiers, and Fractional Graph Colouring

Henning Hasemann
Institute of Operating Systems and Computer Networks,
TU Braunschweig, Germany[1,2]

Juho Hirvonen · Joel Rybicki · Jukka Suomela
Helsinki Institute for Information Technology HIIT, Finland[1,2]
Department of Information and Computer Science,
Aalto University, Finland[1]
Department of Computer Science, University of Helsinki, Finland[2]

[1]current affiliations [2]where most of this work was done

Abstract. In the fractional graph colouring problem, the task is to schedule the activities of the nodes so that each node is active for 1 time unit in total, and at each point of time the set of active nodes forms an independent set.

We show that for any $\alpha > 1$ there exists a deterministic distributed algorithm that finds a fractional graph colouring of length at most $\alpha(\Delta + 1)$ in any graph in one synchronous communication round; here Δ is the maximum degree of the graph. The result is near-tight, as there are graphs in which the optimal solution has length $\Delta + 1$.

The result is, of course, too good to be true. The usual definitions of scheduling problems (fractional graph colouring, fractional domatic partition, etc.) in a distributed setting leave a loophole that can be exploited in the design of distributed algorithms: the size of the local output is not bounded. Our algorithm produces an output that seems to be perfectly good by the usual standards but it is impractical, as the schedule of each node consists of a very large number of short periods of activity.

More generally, the algorithm demonstrates that when we study distributed algorithms for scheduling problems, we can choose virtually any trade-off between the following three parameters: T, the running time of the algorithm, ℓ, the length of the schedule, and κ, the maximum number of periods of activity for any single node. Here ℓ is the objective function of the optimisation problem, while κ captures the "subjective" quality of the solution. If we study, for example, bounded-degree graphs, we can trivially keep T and κ constant, at the cost of a large ℓ, or we can keep κ and ℓ constant, at the cost of a large T. Our algorithm shows that yet another trade-off is possible: we can keep T and ℓ constant at the cost of a large κ.

Keywords: distributed algorithms, fractional domatic partition, fractional graph colouring, local algorithms, unique identifiers.

1 Introduction

In the study of *deterministic distributed algorithms*, it is commonly assumed that there are *unique numerical identifiers* available in the network: in an n-node network, each node is labelled with a unique $O(\log n)$-bit number.

In the general case, numerical identifiers are, of course, very helpful—many fast distributed algorithms crucially depend on the existence of numerical identifiers, so that they can use the Cole–Vishkin technique [2] and similar tricks. However, when we move towards the fastest possible distributed algorithms, the landscape looks very different.

1.1 Local Algorithms and Numerical Identifiers

We focus on *local algorithms* [9, 12], i.e., distributed algorithms that run in constant time (a constant number of communication rounds), independently of the size of the network. In this context, it is no longer obvious if unique identifiers are of any use:

1. In their seminal work, Naor and Stockmeyer [9] prove that there is a class of problems—so-called LCL problems—that do not benefit from unique numerical identifiers: if an LCL problem can be solved with a local algorithm, it can also be solved with an *order-invariant* local algorithm. Order-invariant algorithms do not exploit the numerical value of the identifier; they merely compare the identifiers with each other and use the relative order of the identifiers.

2. More recently, Göös et al. [3] have shown that for a large class of optimisation problems—so-called PO-checkable problems—local algorithms do not benefit from any kind of identifiers: if a PO-checkable optimisation problem can be approximated with a local algorithm, the same approximation factor can be achieved in anonymous networks if we are provided with a port-numbering and an orientation.

While the precise definitions of LCL problems and PO-checkable problems are not important here, they both share the following seemingly technical requirement: it is assumed that the *size of a local output is bounded by a constant* (here the size refers to the number of bits in the encoding of the local output). That is, for each node in the network, there is only a constant number of possible local outputs, independently of the size of the network. However, previously it has not been known whether this is a necessary condition or merely a proof artefact—while contrived counter-examples exist, natural counter-examples have been lacking.

1.2 Contributions

In this work we provide the missing piece of the puzzle: we show that the condition is necessary, even if we focus on natural graph problems and natural encodings of local outputs. More precisely, we show that there is a classical graph problem—namely, *fractional graph colouring* (see Section 2)—with the following properties:

1. In a natural problem formulation, the local outputs can be arbitrarily large.
2. The problem can be solved with a deterministic local algorithm; the algorithm exploits both numerical identifiers and unbounded local outputs.
3. The problem cannot be solved with a deterministic local algorithm without numerical identifiers.
4. The problem cannot be solved with a deterministic local algorithm if we require that the local outputs are of a constant size.

Moreover, this is not an isolated example. The same holds for many other scheduling problems—for example, *fractional domatic partitions* have similar properties (see Section 7). It is up to the reader's personal taste whether this work should be interpreted as a novel technique for the design of local algorithms, or as a cautionary example of a loophole that needs to be closed.

The present work is an extended and revised version of a preliminary conference report [4]. In comparison with the conference version, the material related to fractional domatic partitions is new.

1.3 Comparison with Other Graph Problems

In the study of local algorithms, one often has to make some assumptions on the graph family [5–8]. The most commonly used assumption is to focus on bounded-degree graphs.

If we have a constant maximum degree Δ, then a constant-size local output is a very natural property that is shared by a wide range of combinatorial graph problems—at least if we use a natural encoding of the solution:

1. *Independent set, vertex cover, dominating set, connected dominating sets, etc.*: The output is a subset $X \subseteq V$ of nodes. Each node outputs 1 or 0, indicating whether it is part of X.
2. *Matching, edge cover, edge dominating set, spanning subgraphs, etc.*: The output is a subset $Y \subseteq E$ of edges. A node of degree d outputs a binary vector of length d, with one bit for each incident edge.

3. *Vertex colouring, domatic partition, minimum cut, maximum cut, etc.*: The output is a partitioning of nodes, $X_1 \cup X_2 \cup \cdots \cup X_k = V$. Each node outputs an integer $i \in \{1, 2, \ldots, k\}$, indicating that it belongs to subset X_i. In most cases, there is a natural constant upper bound on k: for example, a vertex colouring does not need more than $\Delta + 1$ colours, a domatic partition cannot contain more than $\Delta + 1$ disjoint dominating sets, and a cut by definition has $k = 2$.

4. *Graph properties*: Each node outputs 1 or 0. For a yes-instance, all nodes have to output 1, and for a no-instance, at least one node has to output 0.

Now if we consider the linear programming (LP) relaxations of problems such as independent sets, vertex covers, or dominating sets, we arrive at a graph problem in which local outputs could be potentially arbitrarily large: each node outputs a rational number, and there is no a priori reason to require that the size of the output (i.e., the length of the binary encoding of the rational number) is bounded. However, it seems that for these problems the size of the output cannot be exploited by a local algorithm—for example, in the case of packing and covering LPs, an exact solution cannot be found by any local algorithm, and the local approximation schemes [7, 8] do not need to exploit unbounded local outputs. Indeed, if we had an algorithm that produces arbitrarily large outputs, we could apply a simple rounding scheme without losing too much in the approximation ratio.

However, fractional graph colouring—the LP relaxation of the vertex colouring problem—is a different story. There we not only have unbounded local outputs, but we show that we can exploit this property in the design of local algorithms.

2 Fractional Graph Colouring

In the fractional graph colouring problem, the task is to coordinate the activities of the nodes in a conflict-free manner. Each node has to perform at least one unit of work, and whenever a node is active all of its neighbours have to be inactive. The objective is to minimise the total length of the schedule, i.e., complete the activities as quickly as possible. The applications include the coordination of radio transmissions in a wireless network: each node must transmit one unit of data, and the transmissions of adjacent nodes interfere with each other.

2.1 Definitions

Let $G = (V, E)$ be a simple, undirected graph that represents a distributed system: each node $v \in V$ is a computational entity, and each edge $\{u, v\} \in E$

represents a communication link between a pair of nodes. Let

$$\mathcal{I} = \{I \subseteq V : \text{if } u, v \in I \text{ then } \{u, v\} \notin E\}$$

consist of all *independent sets* of G. A *fractional graph colouring* associates a value $x(I) \geq 0$ to each $I \in \mathcal{I}$ such that

$$\sum_{I \in \mathcal{I}:\, v \in I} x(I) \geq 1 \quad \text{for all} \quad v \in V.$$

The *length* of a colouring x is

$$\ell(x) = \sum_{I \in \mathcal{I}} x(I),$$

and an optimal fractional graph colouring minimises $\ell(x)$. See Figure 1a for an illustration of a fractional graph colouring, and Figure 2 for a comparison of non-fractional and fractional graph colourings.

The connection between a colouring x and a conflict-free schedule is straightforward: we simply allocate a time slot of length $x(I)$ to I. For example, if we are given a colouring x, we can choose an arbitrary ordering $\mathcal{I} = \{I_1, I_2, \dots\}$ on $\mathcal{I}$, and schedule the activities of the nodes as follows: first all nodes in I_1 are active for $x(I_1)$ time units, then all nodes in I_2 are active for $x(I_2)$ time units, etc.; after $\ell(x)$ time units each node has been active for at least one time unit. Conversely, if we can coordinate the activities, we can construct a graph colouring x, as at each point in time the set of active nodes is in $\mathcal{I}$.

2.2 Schedules of Nodes

When we study fractional graph colouring in a distributed setting, we assume that each node produces its own part of the solution. That is, each node must know when it is supposed to be active. Formally, the *schedule of a node* $v \in V$ is a union of disjoint intervals

$$s(v) = (a_1, b_1] \cup (a_2, b_2] \cup \cdots \cup (a_k, b_k].$$

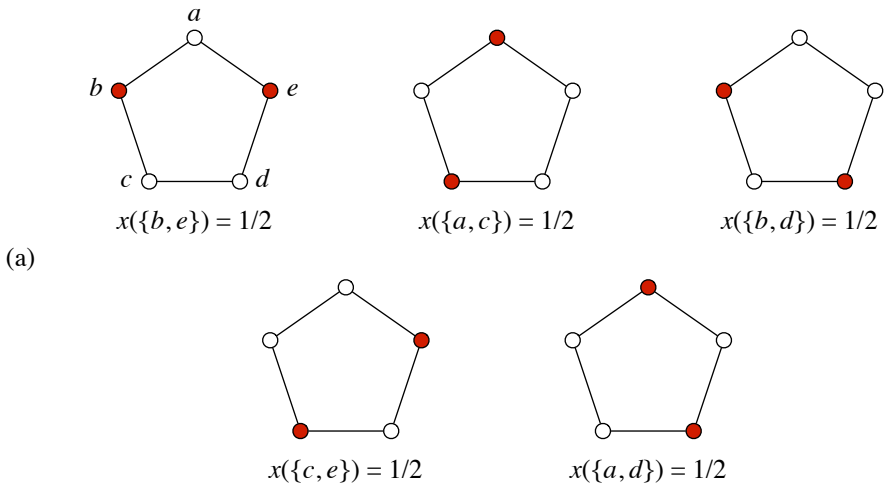
Here $0 \leq a_1 < b_1 < a_2 < b_2 < \cdots < a_k < b_k$ are rational numbers. We require that the total length of the time intervals is at least 1, that is, $\sum_i (b_i - a_i) \geq 1$. The *local output* of node v is the binary encoding of the sequence $a_1, b_1, a_2, b_2, \dots, a_k, b_k$.

We say that node v is active at time t if $t \in s(v)$. Let

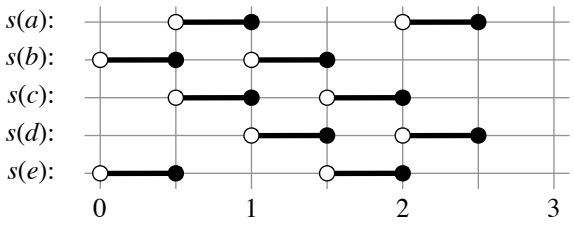
$$A(t, s) = \{v \in V : t \in s(v)\}$$

consist of the nodes that are active at time t. It is straightforward to see that a schedule s defines a fractional graph colouring x of length at most L if

$$A(t, s) = \emptyset \quad \text{for all} \quad t > L,$$
$$A(t, s) \in \mathcal{I} \quad \text{for all} \quad t \leq L.$$



(a)



(b)

Figure 1: (a) A fractional graph colouring x of length $\ell(x) = 5/2$ for the 5-cycle. (b) The schedules of the nodes; each node is active for 1 time unit in total, and no node is active after time $5/2$.

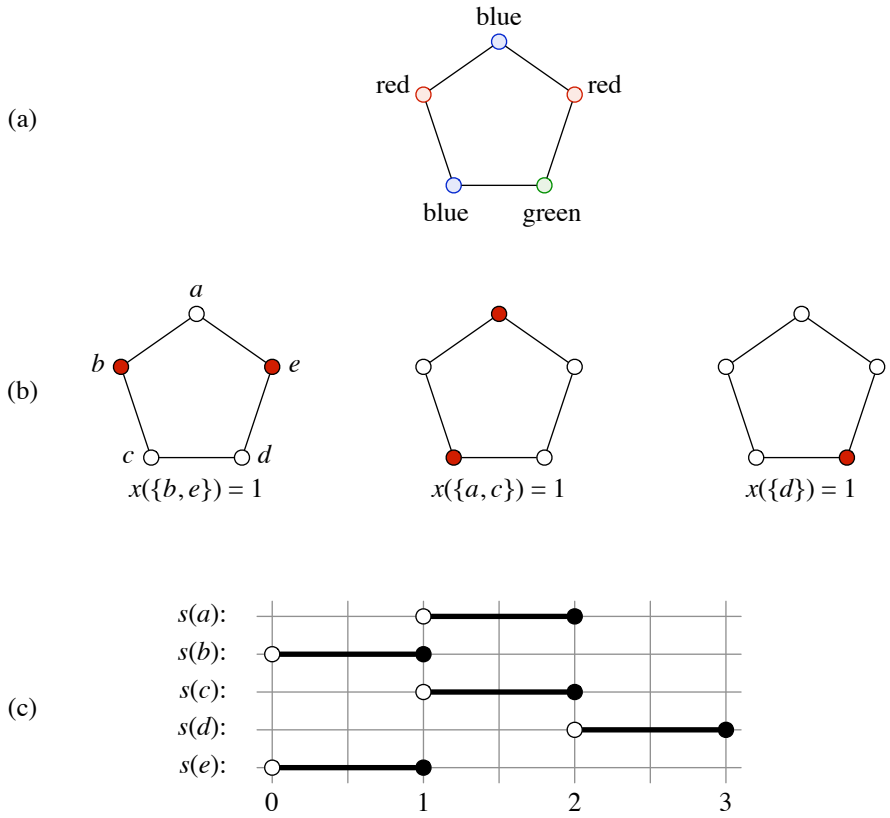


Figure 2: (a) A graph colouring with 3 colours. (b) The same graph colouring interpreted as a fractional graph colouring x of length $\ell(x) = 3$. (c) The schedules of the nodes; each node is active for 1 time unit in total, and no node is active after time 3. The schedule is closely related to the original graph colouring: first all red nodes are active for 1 time unit, then all blue nodes are active for 1 time unit, and finally the green node is active for 1 time unit. Note that here each node is active for only one continuous interval, while in the optimal schedule (see Figure 1) the activities of the nodes are split in multiple intervals.

Equivalently, we have the locally checkable conditions

$$\begin{aligned} \max s(v) &\leq L \quad \text{for each} \quad v \in V, \\ s(u) \cap s(v) &= \emptyset \quad \text{for each} \quad \{u, v\} \in E. \end{aligned}$$

See Figures 1b and 2c for illustrations.

3 Model of Distributed Computing

All of our results hold in the LOCAL model [10]. In this model, we assume that each node $v \in V$ has a unique identifier $f(v) \in \{1, 2, \ldots, \mathsf{poly}(|V|)\}$. Initially, each node knows its own identifier and its degree. Computation proceeds in synchronous communication rounds. In every round, each node in parallel (1) sends a message to each of its neighbours, (2) receives a message from each of its neighbours, (3) updates its own state. After each round, a node can stop and announce its local output. All state transitions are deterministic; there is no source of randomness available. The *running time* is the number of communication rounds until all nodes have stopped. The size of a message is unbounded, and we do not restrict local computation.

To keep the positive result of Theorem 1 as general as possible, we will not use the assumption that we have globally unique identifiers. We only assume that we have some labelling $f \colon V \to \mathbb{N}$ such that $f(u) \neq f(v)$ for each edge $\{u, v\} \in E$. Put otherwise, we only assume that we are given some proper vertex colouring f of G—this is not to be confused with the fractional graph colouring x that we are going to output.

When we turn our attention to fractional domatic partitions in Section 7, we will need a slightly stronger assumption; we will return to this issue in due course.

4 Main Results

Now we are ready to give the main result of this work.

Theorem 1. *For any $\alpha > 1$ there exists a deterministic local algorithm $\mathcal{A}$ such that in any graph G algorithm $\mathcal{A}$ finds a fractional graph colouring x for G in one communication round. Moreover, the length of x is at most $\alpha \cdot (\Delta + 1)$, where Δ is the maximum degree of G.*

We emphasise that algorithm $\mathcal{A}$ does not need to know the number of nodes in G, the maximum degree of G, or any other properties of G. Moreover, the running time is 1, independently of G. However, the theorem heavily abuses the fact that the size of the output is unbounded—in our algorithm, the size of a local output may be superexponential in maximum label, maximum degree, and accuracy $1/(\alpha - 1)$.

The result is near-tight in the sense that there are graphs that do not have a fractional graph colouring of length shorter than $\Delta + 1$. A simple example is the complete graph on $\Delta + 1$ nodes: an optimal fractional graph colouring has length $\Delta + 1$.

From the perspective of the approximability of minimum-length fractional graph colouring, we cannot do much better, either; the following lower bound leaves only a logarithmic gap. Note that the lower bound holds even in the case of d-regular graphs, and even if the running time of the algorithm is allowed to depend on d.

Theorem 2. *Let $\mathcal{F}_d$ be the family of d-regular graphs, and let $\mathcal{A}_d$ be a deterministic algorithm that finds a fractional graph colouring for any $G \in \mathcal{F}_d$ in T_d communication rounds (here T_d is a constant that may dependend on d). Then for each d there is a graph $G_d \in \mathcal{F}_d$ such that G_d admits a fractional graph colouring of length 2, but $\mathcal{A}_d$ outputs a fractional graph colouring of length $\Omega(d/\log d)$.*

Incidentally, in the case of triangle-free graphs, the gap could be closed—we could improve the upper bound by borrowing ideas from Shearer's algorithm [11]. Closing the gap for the case of general graphs is left for future work.

The rest of this paper is structured as follows. We will prove Theorem 1 in Section 5 and Theorem 2 in Section 6. Finally, Section 7 demonstrates that our algorithm design techniques can be extended to other problems as well.

5 Proof of Theorem 1

Informally, our algorithm builds on the following idea: We take an appropriate *randomised* algorithm $\mathcal{A}'$ that produces independent sets. The running time of the randomised algorithm is 1, and it does not require that the random numbers are independent for nodes that are not adjacent. Then we build a deterministic schedule that, essentially, goes through a (very large) number of "random" numbers, and feeds these numbers to $\mathcal{A}'$. Then we simply put together all "random" independent sets that are produced by $\mathcal{A}'$.

The approach is general, in the sense that we could plug in any randomised algorithm $\mathcal{A}'$ that satisfies certain technical properties. However, to keep the presentation readable, we hard-code a specific concrete choice of $\mathcal{A}'$: each node v picks a random number (these are denoted by $w(v, t)$ in Section 5.5), and a node joins the independent set if its number is strictly larger than the numbers picked by its neighbours.

5.1 Preliminaries

Choose $\varepsilon > 0$ and $\beta > 0$ such that

$$\frac{1+\beta}{1-\varepsilon} \leq \alpha.$$

Define $R(x) = \lceil (x+1)/\varepsilon \rceil$. We use the notation $N(v) = \{u \in V : \{u, v\} \in E\}$ for the set of neighbours of $v \in V$, and we write $\deg(v) = |N(v)|$ for the degree of v. Let $N^+(v) = \{v\} \cup N(v)$. The case of an isolated node is trivial; hence we assume that $\deg(v) \geq 1$ for every node v.

5.2 Communication

Recall the definitions of Section 3: we assume that we are given a function f that is a proper vertex colouring of graph $G = (V, E)$. The communication part of the algorithm is nearly trivial: *each node v sends its label $f(v)$ and its degree $\deg(v)$ to each of its neighbours.*

This information turns out to be sufficient to find a fractional graph colouring. The rest of this section explains the local computations that are done by each node; they do not involve any communication at all.

5.3 Scheduling Colours

Let $g\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We say that g is a *scheduling colour function* if

$$\begin{aligned} &g(i,j) \geq j && \text{for all } i \text{ and } j, \\ &g(i,j) \neq g(i',j') && \text{for all } i,\ i',\ j, \text{ and } j' \text{ such that } i \neq i'. \end{aligned}$$

In the algorithm, we will need a scheduling colour function g. For the sake of concreteness, we give an example of such a function (see Table 1):

$$g(i,j) = B(i+j-1) + i - 1, \quad \text{where} \quad B(k) = 2^{\lceil \log_2 k \rceil}.$$

Other choices of g are equally good for our purposes; the choice of g only affects the size of the local outputs. From that perspective it is useful that the values of g are fairly small; with the above definition $g(i,j) = O(i+j)$.

We define that the *scheduling colour* of a node v is

$$c(v) = g\big(f(v), R(\deg(v))\big).$$

We make the following observations:

1. Function $c\colon V \to \mathbb{N}$ is a proper colouring of G, as f was a proper colouring of G.
2. We have $c(v) \geq R(\deg(v))$ for each node v.
3. Each node v knows $c(u)$ for all $u \in N^+(v)$.

1	2	4	4	8	8	8	8	16	16	16	16	16	16	16
3	5	5	9	9	9	9	17	17	17	17	17	17	17	17
6	6	10	10	10	10	18	18	18	18	18	18	18	18	34
7	11	11	11	11	19	19	19	19	19	19	19	19	35	35
12	12	12	12	20	20	20	20	20	20	20	20	36	36	36
13	13	13	21	21	21	21	21	21	21	21	37	37	37	37
14	14	22	22	22	22	22	22	22	22	38	38	38	38	38
15	23	23	23	23	23	23	23	23	39	39	39	39	39	39
24	24	24	24	24	24	24	24	40	40	40	40	40	40	40
25	25	25	25	25	25	25	41	41	41	41	41	41	41	41

Table 1: A scheduling colour function $g(i, j)$, with $i = 1, 2, \ldots$ on rows and $j = 1, 2, \ldots$ on columns. Each integer occurs in at most one row.

5.4 Coordinates

A *coordinate* is a sequence $p = (p_1, p_2, \ldots, p_\ell)$ where $p_i \in \{0, 1, \ldots, i-1\}$. Here ℓ is the *dimension* of the coordinate; we write $\emptyset$ for the coordinate of dimension $\ell = 0$. Note that there are $i!$ coordinates of dimension i.

Define $\beta_i = \beta/(i!)$ for each $i \geq 0$. With each coordinate p of dimension ℓ, we associate a time interval $T(p)$ of length β_ℓ as follows (see Figure 3 for an illustration):

1. For the 0-dimensional coordinate, set $T(\emptyset) = (0, \beta_0]$.
2. Assume that $p = (p_1, p_2, \ldots, p_{i-1})$ is a coordinate of dimension $i-1$ with
$$T\big((p_1, p_2, \ldots, p_{i-1})\big) = (a, a + \beta_{i-1}]$$
for some a. For each $p_i = 0, 1, \ldots, i-1$, we define
$$T\big((p_1, p_2, \ldots, p_i)\big) = \big(a + p_i\beta_i,\ a + (p_i + 1)\beta_i\big].$$

We will use the shorthand notation $T(p_1, p_2, \ldots, p_i)$ for $T\big((p_1, p_2, \ldots, p_i)\big)$.

5.5 First Fragment of the Schedule

Now we are ready to define the schedule within time interval $T(\emptyset)$. To this end, consider a point in time $t \in T(\emptyset)$. Time t defines a unique infinite sequence
$$p(t) = \big(p(1,t), p(2,t), \ldots\big)$$
such that for any i we have
$$t \in T\big(p(1,t), p(2,t), \ldots, p(i,t)\big).$$
Note that $p(k,t)$ ranges over $\{0, 1, \ldots, k-1\}$; see Figure 3 for an illustration.

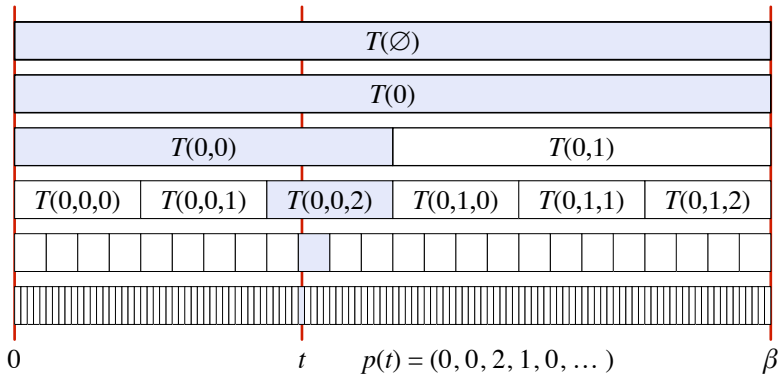


Figure 3: Recursive partitioning $T(p)$ of the interval $(0, \beta]$. Any point in time t defines a unique infinite sequence $p(t) = (p(1,t), p(2,t), \dots)$. If t is picked uniformly at random, then $p(1,t)$, $p(2,t)$, $\dots$ are independent random variables, and $p(i,t)$ is uniformly distributed over $\{0, 1, \dots, i-1\}$. In essence, our partitioning scheme guarantees that we have access to independent, uniformly distributed, discrete random variables of an arbitrarily large range.

We define the *weight* of the colour class $k \in \mathbb{N}$ at time t as follows:

$$W(k,t) = \frac{p(k,t)}{k}.$$

Then we define the weight of a node v at time t as the weight of its scheduling colour:

$$w(v,t) = W(c(v),t).$$

Finally, we define that v is active at time t if it is strictly heavier than any neighbour, that is

$$w(v,t) > w(u,t) \quad \text{for all} \quad u \in N(v). \tag{1}$$

Note that each node v knows $c(u)$ for each $u \in N^+(v)$. Hence each node knows when it is active. Moreover, the schedule can be efficiently computed and it is of finite length. To see this, let

$$c'(v) = \max_{u \in N^+(v)} c(u).$$

Let p be a coordinate of dimension $c'(v)$. Now the weights $w(u,t)$ for $u \in N^+(v)$ are constant during $t \in T(p)$; hence v is either active or inactive during the entire time period $T(c'(v))$. Hence it is sufficient to consider a finite number of time periods.

We will now argue that the schedule for $T(\emptyset)$ is feasible and each node is active for a substantial fraction of $T(\emptyset)$. To this end, define

$$h(v) = \frac{1-\varepsilon}{\deg(v)+1}.$$

Lemma 3. *If $\{u, v\} \in E$, nodes u and v are never active simultaneously during $T(\emptyset)$.*

Proof. This is trivial, as we had a strict inequality in (1). □

Lemma 4. *Each node $v \in V$ is active for at least $\beta h(v)$ time units within time interval $T(\emptyset)$.*

Proof. Assume that we choose a point in time $t \in T(\emptyset)$ uniformly at random. Then the random variables $p(i,t) \in \{0, 1, \ldots, i-1\}$ for $i = 1, 2, \ldots$ are independent and uniformly distributed; it follows that the random variables $W(i,t)$ are also independent and uniformly distributed. For any i and any $0 \le x \le 1$ we have

$$\Pr\big[W(i,t) < x\big] \ge x.$$

Let $v \in V$, and let $C = \{c(u) : u \in N(v)\}$ be the set of scheduling colours in the neighbourhood of v; note that $c(v) \notin C$. Let $n = |C|$ and $k = c(v)$. Summing over all possible values of $W(k,t)$, we have

$$\begin{aligned}
&\Pr\big[\text{node } v \text{ is active at time } t\big] \\
&= \Pr\big[w(v,t) > w(u,t) \text{ for all } u \in N(v)\big] \\
&= \Pr\big[W(k,t) > W(i,t) \text{ for all } i \in C\big] \\
&= \sum_{j=0}^{k-1} \Pr\Big[W(k,t) = \frac{j}{k}\Big] \cdot \Pr\Big[\frac{j}{k} > W(i,t) \text{ for all } i \in C\Big] \\
&\ge \sum_{j=0}^{k-1} \frac{1}{k}\Big(\frac{j}{k}\Big)^n \;=\; \frac{1}{k^{n+1}}\Big(\sum_{j=1}^{k} j^n\Big) - \frac{1}{k} \\
&\ge \frac{1}{k^{n+1}} \int_0^k x^n \, dx - \frac{1}{k} \;=\; \frac{1}{n+1} - \frac{1}{k}.
\end{aligned}$$

Moreover, $n \le \deg(v)$ and $k \ge R(\deg(v)) \ge (\deg(v)+1)/\varepsilon$. Therefore node v is active at time t with probability at least

$$\frac{1}{n+1} - \frac{1}{k} \;\ge\; \frac{1-\varepsilon}{\deg(v)+1} \;=\; h(v). \qquad \square$$

5.6 Complete Schedule

In Section 5.5 we defined the schedule for time interval $T(\emptyset)$. As such, this does not yet constitute a valid fractional graph colouring—indeed, it cannot be the case, as $T(\emptyset)$ is far too short.

However, we can now easily construct a valid solution by repeating the solution that we defined for $T(\emptyset)$. Define

$$H(v) = \left\lceil \frac{1}{\beta h(v)} \right\rceil. \tag{2}$$

Now the schedule $s(v)$ of node v is defined as follows: repeat the schedule defined for $T(\emptyset)$ for $H(v)$ times.

More formally, let $t > 0$. If $t \leq \beta$, we have defined in Section 5.5 whether v is active at time t. Otherwise $t = i\beta + t'$, where $t' \in T(\emptyset)$ and $i \in \mathbb{N}$. If $i \geq H(v)$, node v is inactive. Otherwise node v is active at time t iff it is active at time t'.

Lemma 5. *Each node $v \in V$ is active for at least 1 time unit within time interval $(0, \beta H(v))$.*

Proof. Follows from Lemma 4 and (2). □

Lemma 6. *If the maximum degree of G is Δ, then the length of the schedule is at most $\alpha(\Delta + 1)$.*

Proof. Let $v \in V$. We have

$$\begin{aligned}\beta H(v) &\leq \frac{1}{h(v)} + \beta = \frac{\deg(v) + 1}{1 - \varepsilon} + \beta \\ &\leq \frac{\Delta + 1}{1 - \varepsilon} + \beta \leq \frac{1 + \beta}{1 - \varepsilon}(\Delta + 1) \leq \alpha(\Delta + 1).\end{aligned}$$

That is, after time $\alpha(\Delta + 1)$, node v is no longer active. □

This concludes the proof of Theorem 1—we have designed an algorithm that only needs one communication round, yet it yields a fractional graph colouring of length at most $\alpha(\Delta + 1)$.

6 Proof of Theorem 2

The theorem holds even if f assigns unique identifier from the set $\{1, 2, \ldots, n\}$, where n is the number of nodes in G_d. The proof uses the following lemma.

Lemma 7 (Bollobás [1])**.** *For any given integers $d \geq 3$ and $g \geq 3$, there exists a d-regular graph G with n nodes and girth at least g such that any independent set has size at most $O(n \log(d)/d)$.*

Let $\mathcal{F}$ be the family of d-regular graphs. Let $\mathcal{A}$ be a deterministic algorithm, with running time T, that finds a fractional graph colouring for any graph in $\mathcal{F}$. Now let $G = (V, E)$ be a d-regular graph with girth $g \geq 2T + 1$ obtained from Lemma 7. Each independent set I of G has size at most $c|V| \log(d)/d$, for some constant c. Thus any fractional graph colouring of G has length at least $d/(c \log d)$. Choose a bijection $f \colon V \to \{1, 2, \ldots, |V|\}$.

If we run algorithm $\mathcal{A}$ on G with identifiers given by f, the output is a fractional graph colouring x of length at least $d/(c \log d)$. In particular there must be a node $v^* \in V$ that is active at time $t \geq d/(c \log d)$. Moreover,

the radius-T neighbourhood of v^* is a d-regular tree, as G was a high-girth graph.

Now let $G' = (V', E')$ be the bipartite double cover of G. That is, for each node v of G we have two nodes v_1 and v_2 in G', and for each edge $\{u, v\}$ of G we have two edges $\{u_1, v_2\}$ and $\{u_2, v_1\}$ in G'. There is a covering map $\phi\colon V' \to V$ that maps $v_1 \mapsto v$ and $v_2 \mapsto v$; let $\{v_1^*, v_2^*\} = \phi^{-1}(v^*)$. Graph G' has the following properties.

1. Graph G' is bipartite; therefore there is a fractional graph colouring x' in G' with $\ell(x') = 2$.
2. Graph G' is d-regular; that is, $G' \in \mathcal{F}$.
3. The radius-T neighbourhood of $v_1^* \in V'$ is a d-regular tree.
4. The number of nodes is $|V'| = 2|V|$.

To prove the theorem, it is sufficient to show that we can choose the identifiers for $G' \in \mathcal{F}$ so that $\mathcal{A}$ outputs a fractional graph colouring of length $\Omega(d/\log d)$. To this end, observe that we can choose a bijection $f'\colon V' \to \{1, 2, \ldots, |V'|\}$ so that the radius-T neighbourhood of v_1^* in (G', f') is isomorphic to the radius-T neighbourhood of v^* in (G, f). Now apply $\mathcal{A}$ to (G', f'). By construction, the local output of v_1^* in (G', f') equals the local output of v^* in (G, f); in particular, the length of the schedule x' constructed by $\mathcal{A}'$ is $\Omega(d/\log d)$.

7 Fractional Domatic Partitions

So far we have discussed the fractional graph colouring problem. Now we will turn our attention to another scheduling problem: fractional domatic partitions. While the fractional graph colouring problem was an example of a *minimisation* problem, the fractional domatic partition problem is an example of a *maximisation* problem: we want to make the schedule as long as possible.

7.1 Introduction

Informally, in the fractional domatic partition problem, the task is to coordinate the activities of the nodes so that we maintain a full coverage: if a node is inactive, at least one of its neighbours has to be active. Each node can be active for at most one time unit in total, and the objective is to maximise the total length of the schedule.

One example of an application is lifetime maximisation in a battery-powered sensor network. Our goal is to maximise the lifetime of the system. During the lifetime, all locations need to be monitored—if we turn off a

sensor, at least one adjacent sensor has to be active. Each sensor has a limited battery which provides enough power for one time unit of activity.

7.2 Definitions

Recall that we use the notation $N^+(v)$ for the set that consists of node v and its neighbours. Let

$$\mathcal{D} = \{D \subseteq V : D \cap N^+(v) \neq \emptyset \text{ for each } v \in V\}$$

consist of all *dominating sets* of G. A *fractional domatic partition* associates a value $x(D) \geq 0$ to each $D \in \mathcal{D}$ such that

$$\sum_{D \in \mathcal{D}:\, v \in D} x(D) \leq 1 \quad \text{for all} \quad v \in V.$$

The *length* of a partition x is

$$\ell(x) = \sum_{D \in \mathcal{D}} x(D),$$

and an optimal fractional domatic partition maximises $\ell(x)$.

As before, in a distributed setting, the *schedule of a node* $v \in V$ is a union of disjoint intervals

$$s(v) = (a_1, b_1] \cup (a_2, b_2] \cup \cdots \cup (a_k, b_k].$$

Here $0 \leq a_1 < b_1 < a_2 < b_2 < \cdots < a_k < b_k$ are rational numbers. We require that the total length of the time intervals is at most 1, that is, $\sum_i (b_i - a_i) \leq 1$.

A node v is active at time t if $t \in s(v)$; let $A(t, s) = \{v \in V : t \in s(v)\}$ consist of the nodes that are active at time t. A schedule s defines a fractional domatic partition x of length at least L if

$$A(t, s) \in \mathcal{D} \quad \text{for all} \quad t \leq L.$$

Equivalently, we have the locally checkable condition

$$(0, L] \subseteq \bigcup_{u \in N^+(v)} s(u) \quad \text{for each} \quad v \in V.$$

7.3 Model

It turns out that we cannot prove a result analogous to Theorem 1 for fractional domatic partitions without slightly strengthening the assumptions. In Section 3 we assumed that f is a proper colouring of G, and this turned out to enough to solve the fractional graph colouring problem.

However, this is no longer the case with fractional domatic partitions. To see this, consider, for example, the complete bipartite graph $G = K_{n,n}$, which admits a fractional domatic partition of length at least n. However, G can be coloured with two colours, and given G and a 2-colouring f of G, a deterministic distributed algorithm cannot find a fractional domatic partition of length larger than 2; in essence, we cannot break the symmetry between the nodes of the same colour class.

Therefore we will assume in this section that f is a proper *distance*-2 *colouring* of G. That is, if $u \neq v$ are adjacent nodes, or they share a common neighbour, then $f(u) \neq f(v)$. Put otherwise, for any node $v \in V$ we have

$$|\{f(u) : u \in N^+(v)\}| = \deg(v) + 1. \tag{3}$$

We are primarily interested in the case of unique identifiers, and in that case (3) is trivially satisfied.

We will also assume, without loss of generality, that there are *no isolated nodes* in G. Indeed, if there is an isolated node, the length of any fractional domatic partition is at most 1, and the optimal schedule of an isolated node is trivial to find.

7.4 Result

The main result of this section is summarised in the following theorem.

Theorem 8. *There exists a deterministic local algorithm that finds a fractional domatic partition x for any graph G in one communication round. Moreover, the length of x is at least*

$$\frac{\delta + 1}{3 \ln(\delta + 1)},$$

where $\delta \geq 1$ is the minimum degree of graph G.

Note that *any* fractional domatic partition has length at most $\delta + 1$. Therefore our algorithm is guaranteed to find an $O(\log(\delta+1))$-approximation of an optimal fractional domatic partition. Again, we have only a logarithmic gap between the upper and lower bounds.

The constant 3 in the statement of Theorem 8 is in no way magical; we just picked a nice round number that makes the proof easy to follow and avoids a tedious analysis of certain corner-cases. A more careful analysis would yield a slightly tighter value.

7.5 Overview

The rest of this section presents the proof of Theorem 8. The general approach is familiar from Section 5: We have a *randomised* algorithm $\mathcal{A}'$ that finds a

dominating set in one communication round. Then a *deterministic* algorithm feeds appropriate "random" numbers to $\mathcal{A}'$ to construct a fractional domatic partition—each point of time corresponds to one possible assignment of random numbers.

In Section 5, the randomised algorithm was nearly trivial: at time t, each node v picks a random weight $w(v,t)$, and a node joins the independent set if it is strictly heavier than any of its neighbours.

In the case of fractional domatic partitions, the design of the randomised algorithm requires more care. Therefore we will first design and analyse the randomised algorithm and only after that explain how to use it as a black box to construct an appropriate deterministic schedule.

7.6 Preliminaries

The following expressions appear so frequently that we will define a shorthand notation:

$$\lambda(x) = \frac{x+1}{\ln(x+1)}, \quad \Lambda(x) = \frac{\ln(x+1)}{x+1}.$$

With this notation, our algorithm produces a schedule of length at least $\lambda(\delta)/3$. Note that $\lambda(x)$ is monotonically increasing and $\Lambda(x)$ is monotonically decreasing for $x \geq 2$.

We fix the constants $\varepsilon = 0.9$ and $\beta = 1/60$. We have chosen the values of ε and β so that they satisfy

$$3(1-2\beta) = 2 + \varepsilon = 2.9 > \lambda(1).$$

Informally, the constant $\lambda(1)$ is related to the trivial case of $\delta = 1$, while the constant $2 + \varepsilon = 2.9$ will appear in the analysis of the non-trivial case. This leaves us some slack in comparison with the constant 3 that appears in the statement of Theorem 8.

Let $g\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a scheduling colour function (recall Section 5.3). In our algorithm, we will need the following values that are derived from the degrees of the nodes:

$$\begin{aligned} \tilde{d}(v) &= \max\{\, 2, \deg(v) \,\}, \\ \gamma(v) &= \min_{v \in N^+(v)} \tilde{d}(v), \\ \gamma(v,u) &= \min\{\tilde{d}(v), \tilde{d}(u)\}. \end{aligned}$$

We also define two functions that discretise λ and Λ:

$$\begin{aligned} S(x) &= \lceil \lambda(x)/\varepsilon \rceil, \\ Q(x,k) &= \lceil k\Lambda(x) \rceil / k. \end{aligned}$$

Now the *scheduling colour* of a node v is

$$c(v) = g\big(f(v), S(\tilde{d}(v))\big).$$

In our randomised algorithm, the probability that a node decides to act is related to the following expressions:

$$q(v) = Q\big(\gamma(v), c(v)\big),$$
$$q(v,u) = Q\big(\gamma(v,u), c(u)\big).$$

Note that we have $q(v) \in C(v)$ and $q(v,u) \in C(u)$, where

$$C(v) = \{i/c(v) : i = 0, 1, \ldots, c(v) - 1\}.$$

7.7 Randomised Algorithm

Informally, we would like to apply a straightforward randomised algorithm that proceeds as follows. Here we will use the words "join", "dominate", and "cover" to describe the internal state of a node, the idea being that the nodes in state "join" together with the nodes in state "cover" will form a dominating set:

1. Node v *joins* with probability $q(v)$.
2. Node v *dominates* a neighbour $u \in N(v)$ whenever it joins.
3. Node v *covers* if it does not join and none of its neighbours dominates it.
4. Node v is active if it either joins or covers.

Unfortunately, a direct implementation of this scheme is not possible in one communication round. To see this, consider a path (v_1, v_2, v_3) of length two. Now the decision of v_3 (does it need to cover) depends on $q(v_2)$, which depends on $\deg(v_1)$.

Therefore we will slightly adjust the algorithm:

1. Node v picks a weight $w(v) \in C(v)$ uniformly at random. We will assume that for each $v \in V$, the random variables $w(u)$ for $u \in N^+(v)$ are mutually independent.
2. Node v *joins* if $w(v) < q(v)$.
3. Node v *dominates* a neighbour $u \in N(v)$ if $w(v) < q(u,v)$.
4. Node v *covers* if it does not join and none of its neighbours dominates it.
5. Node v is active if it either joins or covers.

It turns out that this scheme has all the properties that we need. The key observations are summarised in the following lemmas.

Lemma 9. *The decision of a node $v \in V$ only depends on the following values:* $\deg(u)$, $f(u)$, *and* $w(u)$ *for* $u \in N^+(v)$.

Proof. The data flows are illustrated in Figure 4. □

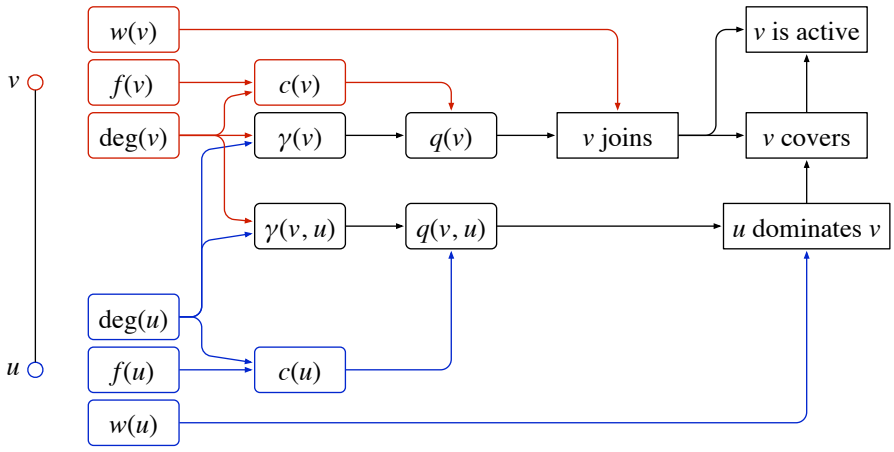


Figure 4: The decision of whether v is active only depends on the data available at v and at its neighbours $u \in N(v)$. Note that v will not know whether u joins; it will only know whether u dominates v.

Lemma 10. *The set of active nodes forms a dominating set.*

Proof. Let $v \in V$. Assume that v is not active; we need to prove that at least one node $u \in N(v)$ is active.

As v is not active, there is a node $u \in N(v)$ that dominates v. By definition, we have $w(u) < q(v,u)$. To prove that u is active, it is sufficient to show that $w(u) < q(v,u) \leq q(u)$.

To this end, first observe that

$$2 \leq \gamma(u) \leq \gamma(v,u).$$

Therefore

$$\Lambda(\gamma(u)) \geq \Lambda(\gamma(v,u)),$$

which implies

$$q(u)c(u) = \lceil c(u)\Lambda(\gamma(u)) \rceil \geq \lceil c(u)\Lambda(\gamma(v,u)) \rceil = q(v,u)c(u).$$

Hence $w(u) < q(v,u) \leq q(u)$; node u is active. □

Lemma 11. *A node v joins with probability at most $(1+\varepsilon)\Lambda(\gamma(v))$.*

Proof. We have

$$Q(x,k) < \Lambda(x) + 1/k,$$

$$c(v) \geq S(\tilde{d}(v)) \geq \lambda(\tilde{d}(v))/\varepsilon \geq \lambda(\gamma(v))/\varepsilon,$$

which implies

$$\Pr[v \text{ joins}] = q(v) = Q\big(\gamma(v), c(v)\big) < (1+\varepsilon)\Lambda(\gamma(v)).$$

The first equality follows from the fact that both $w(v)$ and $q(v)$ are multiples of $1/c(v)$. □

Lemma 12. *A node v with $\deg(v) \geq 2$ covers with probability at most $1/(\gamma(v)+1)$.*

Proof. The probability that v does not join is $1 - q(v)$. The probability that a neighbour $u \in N(v)$ does not dominate v is $1 - q(v,u)$; moreover, all of these events are mutually independent. Therefore the probability that v covers is

$$\Pr[v \text{ covers}] = (1 - q(v)) \prod_{u \in N(v)} (1 - q(v,u)).$$

Now we have

$$\begin{aligned} q(v) &\geq \Lambda(\gamma(v)), \\ q(v,u) &\geq \Lambda(\gamma(v,u)) \geq \Lambda(\gamma(v)), \\ |N(v)| &= \deg(v) = \tilde{d}(v) \geq \gamma(v). \end{aligned}$$

Hence

$$\Pr[v \text{ covers}] \leq \big(1 - \Lambda(\gamma(v))\big)^{\gamma(v)+1} = \left(1 - \frac{\ln(\gamma(v)+1)}{\gamma(v)+1}\right)^{\gamma(v)+1}.$$

We apply the inequality

$$\left(1 + \frac{\ln x}{n}\right)^n \leq x$$

with $x^{-1} = n = \gamma(v) + 1$ to derive

$$\Pr[v \text{ covers}] \leq \frac{1}{\gamma(v)+1}.$$ □

Lemma 13. *A node v is active with probability at most $(2+\varepsilon)\Lambda(\delta)$.*

Proof. If $\delta = 1$, the claim is vacuous:

$$(2+\varepsilon)\Lambda(1) > 1.005.$$

Otherwise $\delta \geq 2$, which implies $\deg(v) \geq \gamma(v) \geq \delta \geq 2$. We have from Lemmas 11 and 12

$$\Pr[v \text{ is active}] \leq (1+\varepsilon)\Lambda(\gamma(v)) + \frac{1}{\gamma(v)+1}.$$

As $\ln(\gamma(v)+1) \geq 1$, we have

$$\Pr[v \text{ is active}] \leq (2+\varepsilon)\Lambda(\gamma(v)) \leq (2+\varepsilon)\Lambda(\delta).$$ □

7.8 Schedule

Now we will use the randomised algorithm of Section 7.7 as a black box to design a deterministic algorithm for fractional domatic partitions.

We borrow the definitions of *coordinates* p and *time intervals* $T(\cdot)$ from Section 5.4. Following Section 5.5, a point of time $t \in T(\emptyset)$ defines a unique infinite sequence

$$p(t) = \big(p(1,t), p(2,t), \dots\big).$$

We use this sequence to define the *weight* of a node v at time t as follows:

$$W(k,t) = \frac{p(k,t)}{k},$$
$$w(v,t) = W(c(v),t).$$

If we pick a point of time $t \in T(\emptyset)$ uniformly at random, then from the perspective of each node v, the weights $w(u,t)$ for all $u \in N^+(v)$ are independent random variables; moreover, each $w(v,t)$ is picked uniformly at random from $C(v)$.

Now we simply simulate the randomised algorithm of Section 7.7; for each node v, we assume that the random weight $w(v)$ is equal to $w(v,t)$. Lemma 9 implies that each node v can determine its own schedule after one communication round. Lemma 10 implies that at any point of time $t \in T(\emptyset)$, the set of active nodes forms a dominating set, and Lemma 13 implies that during the time interval $T(\emptyset) = (0, \beta]$, each node is active for at most $\beta(2+\varepsilon)\Lambda(\delta)$ time units.

So far we have defined the schedule of each node for the time interval $T(\emptyset) = (0, \beta]$. To construct a fractional domatic partition x, we simply repeat the schedule that we defined for $T(\emptyset)$, as long as possible. Conceptually, each node first constructs an infinitely long schedule, and then takes the longest prefix during which it is active for at most 1 time unit in total. Different nodes may stop at different points; however, none of the nodes stop during the first

$$N = \left\lfloor \frac{\lambda(\delta)}{\beta(2+\varepsilon)} \right\rfloor \geq \frac{\lambda(\delta)}{\beta(2+\varepsilon)} - 1$$

time intervals of length β. Now observe that

$$\frac{\lambda(\delta)}{2+\varepsilon} > \frac{1}{2}$$

for any $\delta \geq 1$. Therefore the total length of the fractional domatic partition x is at least

$$\beta N \geq \frac{\lambda(\delta)}{2+\varepsilon} - \beta > \frac{(1-2\beta)\lambda(\delta)}{2+\varepsilon} = \frac{\lambda(\delta)}{3}.$$

This completes the proof of Theorem 8.

8 Discussion

We have shown that the fractional graph colouring problem and the fractional domatic partition problem can be solved very quickly in a distributed setting—if and only if we do not impose restrictions on the size of the local outputs.

More generally, we can approach scheduling problems from the following perspective. We have three parameters:

1. T, the running time of the distributed algorithm,
2. ℓ, the length of the schedule (objective function),
3. κ, the maximum number of disjoint time intervals in the schedule of a node.

Now for the sake of concreteness, let us focus on the case of fractional graph colouring on bounded-degree graphs, i.e., $\Delta = O(1)$. Our work shows that we can keep any two of T, ℓ, and κ constant, but not all three of them:

1. $T = O(1)$ and $\kappa = O(1)$: trivial, set $s(v) = (f(v), f(v) + 1]$.
2. $\kappa = O(1)$ and $\ell = O(1)$: easy, find an $O(1)$-colouring c and set $s(v) = (c(v), c(v) + 1]$.
3. $T = O(1)$ and $\ell = O(1)$: possible, using Theorem 1.
4. $T = O(1)$, $\ell = O(1)$, and $\kappa = O(1)$: impossible. Now we have an LCL-problem. It is easy to see that the problem cannot be solved with an order-invariant local algorithm (consider a cycle), and hence the result by Naor and Stockmeyer [9] implies that the problem cannot be solved with any local algorithm.

Acknowledgements

This work is an extended and revised version of a preliminary conference report [4]. We thank the anonymous reviewers for their helpful feedback and comments. This work has been partially supported by the European Union under contract number ICT-2009-258885 (SPITFIRE), the Academy of Finland, Grants 132380 and 252018, the Research Funds of the University of Helsinki, and the Finnish Cultural Foundation.

References

[1] Béla Bollobás. The independence ratio of regular graphs. *Proceedings of the American Mathematical Society*, 83(2):433–436, 1981.

[2] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1): 32–53, 1986. doi:10.1016/S0019-9958(86)80023-7.

[3] Mika Göös, Juho Hirvonen, and Jukka Suomela. Lower bounds for local approximation. In *Proc. 31st Annual ACM Symposium on Principles of Distributed Computing (PODC 2012)*, pages 175–184. ACM Press, 2012. doi:10.1145/2332432.2332465. arXiv:1201.6675.

[4] Henning Hasemann, Juho Hirvonen, Joel Rybicki, and Jukka Suomela. Deterministic local algorithms, unique identifiers, and fractional graph colouring. In *Proc. 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2012)*, volume 7355 of *Lecture Notes in Computer Science*, pages 48–60. Springer, 2012. doi:10.1007/978-3-642-31104-8_5.

[5] Juho Hirvonen and Jukka Suomela. Distributed maximal matching: greedy is optimal. In *Proc. 31st Annual ACM Symposium on Principles of Distributed Computing (PODC 2012)*, pages 165–174. ACM Press, 2012. doi:10.1145/2332432.2332464. arXiv:1110.0367.

[6] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 300–309. ACM Press, 2004. doi:10.1145/1011767.1011811.

[7] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, pages 980–989. ACM Press, 2006. doi:10.1145/1109557.1109666.

[8] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: lower and upper bounds, 2010. arXiv:1011.5470.

[9] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

[10] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[11] James B. Shearer. A note on the independence number of triangle-free graphs. *Discrete Mathematics*, 46(1):83–87, 1983. doi:10.1016/0012-365X(83)90273-X.

[12] Jukka Suomela. Survey of local algorithms. *ACM Computing Surveys*, 45(2):24:1–40, 2013. doi:10.1145/2431211.2431223. http://www.cs.helsinki.fi/local-survey/.