# Analysing local algorithms in location-aware quasi unit-disk graphs

Marja Hassinen, Joel Kaasinen, Evangelos Kranakis, Valentin Polishchuk, Jukka Suomela, and Andreas Wiese

June 22, 2011

MH, JK, VP, JS:
Helsinki Institute for Information Technology HIIT, University of Helsinki
*Email:* marja.k.hassinen@gmail.com, joel.kaasinen@cs.helsinki.fi, valentin.polishchuk@cs.helsinki.fi, jukka.suomela@cs.helsinki.fi

EK:
School of Computer Science, Carleton University
*Email:* kranakis@scs.carleton.ca

AW:
Institut für Mathematik, Technische Universität Berlin
*Email:* wiese@math.tu-berlin.de

**Abstract.** A local algorithm with local horizon $r$ is a distributed algorithm that runs in $r$ synchronous communication rounds; here $r$ is a constant that does not depend on the size of the network. As a consequence, the output of a node in a local algorithm only depends on the input within $r$ hops from the node.

We give tight bounds on the local horizon for a class of local algorithms for combinatorial problems on unit-disk graphs (UDGs). Most of our bounds are due to a refined analysis of existing approaches, while others are obtained by suggesting new algorithms. The algorithms we consider are based on network decompositions guided by a rectangular tiling of the plane. The algorithms are applied to matching, independent set, graph colouring, vertex cover, and dominating set.

We also study local algorithms on quasi-UDGs, which are a popular generalisation of UDGs, aimed at more realistic modelling of communication between the network nodes. Analysing the local algorithms on quasi-UDGs allows one to assume that the nodes know their coordinates only approximately, up to an additive error. Despite the localisation error, the quality of the solution to problems on quasi-UDGs remains the same as for the case of UDGs with perfect location awareness. We analyse the increase in the local horizon that comes along with moving from UDGs to quasi-UDGs.

## 1 Introduction

Rapid growth of real-world ad-hoc and sensor networks calls for designing efficient distributed algorithms on large-scale networks, in which each node produces its output based only on the information available within a constant number of hops; such algorithms are called *local*. In a purely combinatorial setting, Linial's [40] lower bound immediately takes away any hope of designing a local algorithm for problems such as maximal independent set or graph colouring; it is not even possible to find a nontrivial approximation for problems such as dominating set, independent set, and matching with local algorithms [9, 30, 31, 33, 38, 41]. While there are local algorithms for linear programs [15–18, 30, 33, 34, 43], relatively few deterministic constant-time algorithms are known for classical combinatorial problems. The positive examples include algorithms for vertex covers [1, 2, 30, 33, 41, 46] and algorithms for special cases of dominating sets [9, 36, 37, 39, 49]; see the survey [50] for details.

The situation is different in a geometric setting, when the nodes reside in the plane and each node knows its coordinates (so-called *location-aware* nodes): one can use the coordinates to break the symmetry. Indeed, quite a few local algorithms are known for location-aware graphs [6, 10–12, 19, 20, 23, 25–27, 30, 48, 51, 52, 54–59]. In this work, we give a unified description and analysis of a class of local algorithms based on a simple "tile-and-combine" idea: decompose the plane into tiles, have each tile solve its subproblem optimally, and combine the solutions into a global output.

### 1.1 Model of distributed computing

Let $\mathcal{G} = (V, E)$ be a given undirected graph representing communication between devices in a distributed system: each node $v \in V$ is a device, and each undirected edge $\{u, v\} \in E$ is a bidirectional communication link between the devices. We study the case where $\mathcal{G}$ is a geometric graph: each node $v \in V$ is associated with a point $p(v) \in \mathbb{R}^2$. Furthermore, we assume that the network is location-aware: each node knows its coordinates.

A local algorithm with *local horizon* $r$ consists of $r$ synchronous communication rounds. During each round, every node performs local computations and exchanges messages with its neighbours. We use the model of Linial [40] and Naor and Stockmeyer [42]: the message size is unbounded and local computation is free; Peleg [45] calls this the *local* model. Hence our results are *communication complexity bounds* – what amount of local information is sufficient to solve certain computational problems.

In a local algorithm, the output of a node $v \in V$ depends on the input only at the nodes within $r$ or fewer edges (hops) from $v$; denote the set of such nodes by $B_{\mathcal{G}}(v, r)$. A local algorithm is robust to changes in the network; any changes outside $B_{\mathcal{G}}(v, r)$ do not influence the computation at the node $v$.

In this paper, we investigate the numerical value of the constant $r$ for a family of local algorithms.

## 1.2 Quasi unit-disk graphs

Throughout this work, we assume that the communication graph $\mathcal{G}$ is a *quasi unit-disk graph* (qUDG) [5, 35]. A graph $\mathcal{G} = (V, E)$ is a $d$-qUDG if for $u, v \in V$, $\|p(u) - p(v)\| > 1$ implies $\{u, v\} \notin E$, and $\|p(u) - p(v)\| \leq d$ implies $\{u, v\} \in E$. Here $0 < d \leq 1$ is a constant parameter, and $\|x - y\|$ is the distance between the points $x, y \in \mathbb{R}^2$.

For $d = 1$, a $d$-qUDG is a unit-disk graph (UDG). In a UDG, the coordinates of the nodes determine the edges; in a $d$-qUDG for $d < 1$ this is not true. A qUDG models real-world wireless networks well: two nodes can always communicate if they are close, and never if they are far. For moderate distances (between $d$ and 1), it is not known in advance whether a communication link will be established, as it depends on the subtleties of radio propagation [22, 29].

Another motivation for considering qUDGs is that it allows us to lift the assumption of perfect location awareness. Indeed, assume that $\mathcal{G}$ is a $d$-qUDG, and for some $\epsilon < d/2$ each node $v$ knows an estimate $\hat{p}(v)$ of its true coordinates $p(v)$ with $\|\hat{p}(v) - p(v)\| \leq \epsilon$. Then $\mathcal{G}$ with the embedding $\hat{p}(v)/(1 + 2\epsilon)$ is a $D$-qUDG for $D = (d - 2\epsilon)/(1 + 2\epsilon)$.

Moreover, positive results for qUDGs directly extend to other families of graphs, e.g., to *civilised graphs* [13, §8.5]. In a civilised graph, the minimum distance between nodes is bounded from below; thus, a civilised graph is a qUDG. Civilised graphs are particularly appealing from the perspective of local algorithms: for a constant $r$, the size of $B_{\mathcal{G}}(v, r)$ is bounded by a constant.

## 1.3 Induced subgraphs

Let us first define the *midpoint* $p(e)$ of an edge $e \in E$ by setting $p(e) = (p(u) + p(v))/2$. Now each node $v$ is associated with a point $p(v)$, and each edge $e$ is also associated with a point $p(e)$.

Let $A \subset \mathbb{R}^2$. We define the subgraphs $\mathcal{G}[A]$, $\mathcal{G}[[A]]$, and $\mathcal{G}[A^+]$ of $\mathcal{G}$ as follows (see Figure 1):

- $\mathcal{G}[A] = (V[A], E[A])$ is the subgraph induced by the nodes in $A$;
- $\mathcal{G}[[A]] = (V[[A]], E[[A]])$ is $\mathcal{G}[A]$ augmented with the edges that have only one endpoint in $A$;
- $\mathcal{G}[A^+] = (V[A^+], E[A^+])$ is the subgraph induced by the edges with the midpoint in $A$.

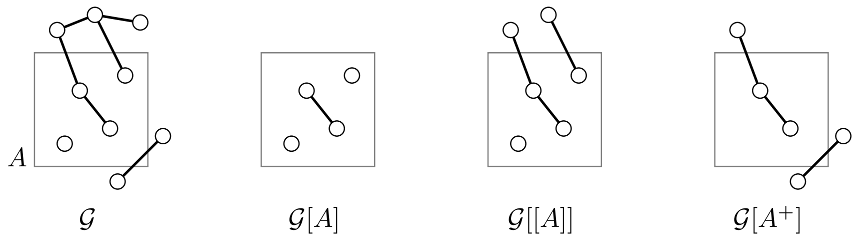

Figure 1: $\mathcal{G}[A]$, $\mathcal{G}[[A]]$, and $\mathcal{G}[A^+]$.

The diameters of $\mathcal{G}[A]$ and $\mathcal{G}[[A]]$ are intimately connected to the local horizons of the algorithms that we study; we make this connection precise in Section 3. The following definitions are central to our analysis.

**Definition 1.** Let $A$ be an $a \times b$ rectangle. $D_{a\times b}(d)$ is the maximum possible diameter of a connected component of $\mathcal{G}[A]$ over all $d$-qUDGs $\mathcal{G}$.

**Definition 2.** Let $A$ be an $a \times b$ rectangle. $E_{a\times b}(d)$ is the maximum possible diameter of a connected component of $\mathcal{G}[[A]]$ over all $d$-qUDGs $\mathcal{G}$.

## 1.4 Contributions

We analyse the local horizon of several tile-and-combine local algorithms for combinatorial problems on qUDGs. Some of the algorithms were known before; we give considerably better bounds for them. We also present and analyse several new algorithms.

In Section 3, we give a unified treatment of the algorithms, expressing their local horizons in terms of $D_{a\times b}(d)$ and $E_{a\times b}(d)$ for relevant values of $a$ and $b$. In Section 5, we present upper bounds for $D_{a\times b}(d)$ and $E_{a\times b}(d)$. In Section 6 we show that many of our bounds are tight or near-tight. In particular, we prove that

$$
\begin{aligned}
D_{1\times1}(1) &= 5, & D_{1\times1}(d) &= 7 & 0.708 \le d &< 1, & (1)\\
D_{2\times1}(1) &= 9, & D_{2\times1}(d) &= 11 & 0.834 \le d &< 1, & (2)\\
D_{2\times2}(1) &= 15, & D_{2\times2}(d) &= 17 & 0.843 \le d &< 1, & (3)\\
D_{3\times2}(1) &= 21, & D_{3\times2}(d) &= 23 & 0.919 \le d &< 1, & (4)\\
D_{4\times2}(1) &= 27, & D_{4\times2}(d) &\ge 29 & d &< 1, & (5)\\
23 \le E_{2\times2}(1) &\le 24, & & & & & (6)\\
39 \le E_{4\times2}(1) &\le 42. & & & & & (7)
\end{aligned}
$$

For UDGs ($d = 1$), our results lead to local algorithms with much better horizons than those presented in prior work (Table 1). For example, for 3-approximate dominating set, a new analysis of an existing algorithm improves the horizon from 83 to 42. For 3-approximate vertex cover, a combination

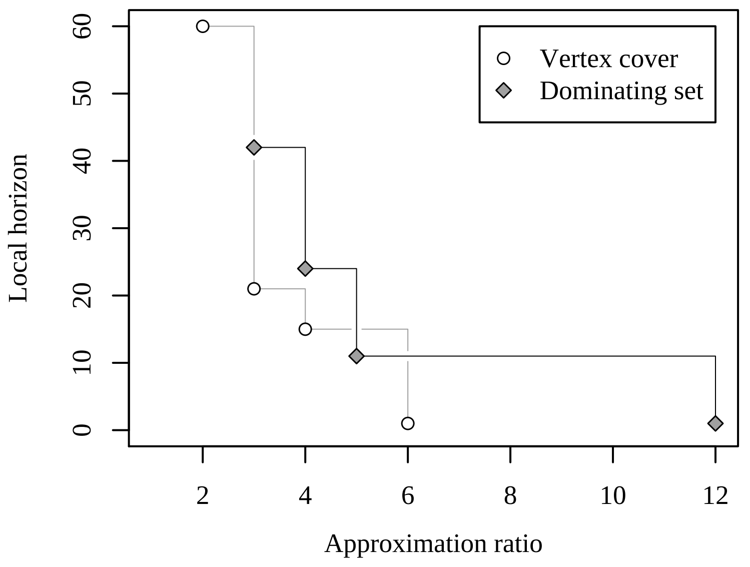

Figure 2: The approximation ratios and the local horizons for vertex cover and dominating set.

of a new algorithm and a new analysis improves the horizon from 83 to 24. Figure 2 shows the graph of Pareto optimal pairs of approximation ratio and local horizon for vertex cover and dominating set problems.

### 1.5 Prior work

The algorithms that we study are based on simple rectangular tilings of the plane; see Sections 2 and 3 for details. A tiling determines a *network decomposition*, given which, one can use a simple "greedy" algorithm to find a maximal independent set and vertex $(\Delta + 1)$-colouring [4]. The greedy algorithms can be also interpreted as an application of the algorithm for *$t$-oriented graphs* [3]. The "parallel" algorithm for approximate vertex colouring in a network decomposition is mentioned by, e.g., Kuhn [30, §1.3.2].

Rectangular tilings have been applied to designing local algorithms for the following problems on UDGs: vertex colouring [54, 57], edge colouring [54], dominating set [23, 54], vertex cover [23, 54, 56], independent set [54], and matching [56]. Variations on the same theme include Kuhn [30, §5.4.1], Moscibroda [41, §8.3.1], and Kuhn et al. [32].

Upper bounds on the local horizons have been given in earlier works; see Table 1 for a summary. In addition to those listed in the table, there is a local approximation algorithm for edge colouring with approximation guarantee $3 \cdot \mathrm{OPT} + 3$ [54]; the local horizon of the algorithm is 260. There are also local approximation schemes for dominating set, vertex cover, and

| Problem | Local horizon | Local horizon, UDGs | |
|---|---|---|---|
| | | This work | Prior work |
| Maximal matching | $4D_{2\times 2}$ | 60 | 381 (a) |
| Maximal independent set | $4D_{1\times 1}+3$ | 23 | |
| 4-approx. independent set | $2E_{4\times 2}+D_{4\times 2}$ | 111 | 211 (b) |
| Vertex $(\Delta+1)$-colouring | $4D_{1\times 1}+3$ | 23 | |
| 3-approx. edge colouring | $D_{3\times 2}$ | 21 | |
| 4-approx. edge colouring | $D_{2\times 2}$ | 15 | |
| 3-approx. vertex colouring | $D_{2\times 1}$ | 9 | 42 (c) |
| 4-approx. vertex colouring | $D_{1\times 1}$ | 5 | |
| 2-approx. vertex cover | $4D_{2\times 2}$ | 60 | 381 (a) |
| 3-approx. vertex cover | $D_{3\times 2}$ | 21 | 83 (e) |
| 4-approx. vertex cover | $D_{2\times 2}$ | 15 | |
| 6-approx. vertex cover | — | — | 1 (h) |
| 3-approx. dominating set | $E_{4\times 2}$ | 42 | 83 (d) |
| 4-approx. dominating set | $E_{2\times 2}$ | 24 | |
| 5-approx. dominating set | — | — | 11 (g) |
| 12-approx. dominating set | — | — | 1 (f) |

(a) [56, Section 4] – rectangular tiling.
(b) [54, Section 5.4] – rectangular tiling.
(c) [54, Section 6.2], [57, Section 3.3] – rectangular tiling.
(d) [23, Section II], [54, Section 5.2] – rectangular tiling.
(e) [23, Section III], [54, Section 5.3] – rectangular tiling.
(f) [54, Section 4.2], [55, Section 2.2] – hexagonal tiling.
(g) [10, Section 2] – hexagonal tiling.
(h) [55, Section 5.2] – no tiling.

Table 1: The algorithms studied in this work, their local horizons in terms of $D_{a\times b}$ and $E_{a\times b}$, and comparison with prior work.

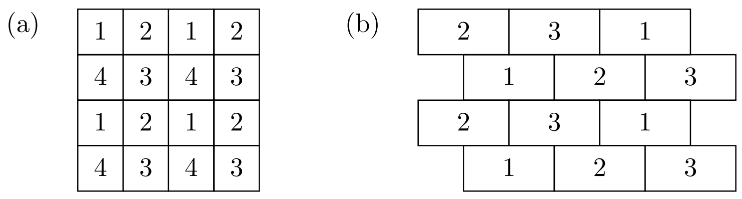

Figure 3: Tilings and colours.

independent set [54, 58, 59]; however, in order to obtain approximation ratios better than 4, the local horizon of these algorithms is larger than 1000.

Few works consider local algorithms for generalisations of UDGs [6, 23, 27]. Kuhn, Moscibroda, and Wattenhofer [30, 32, 41] study qUDGs and their generalisations, but the algorithms are not strictly local, that is, the local horizon depends on the number of nodes in the network.

### 1.6 Roadmap

This paper is organised as follows: Section 2 explains how a node in the network can gather local information needed to produce its local output. Section 3 lists the algorithms and Section 4 argues their correctness. Sections 5 and 6 give upper and lower bounds on $D_{a\times b}$ and $E_{a\times b}$. Section 7 concludes the paper.

## 2 Preliminaries

A particularly simple and effective way to design local algorithms for combinatorial problems on qUDGs is to apply the tilings presented in Figure 3. In Figure 3a the tiles are squares, and each tile is coloured by one of 4 colours. Depending on the algorithm, the dimensions of the tiles are $1 \times 1$ or $2 \times 2$; we refer to these tilings as the 4-*coloured* $1 \times 1$ *tiling* and the 4-*coloured* $2 \times 2$ *tiling*. In Figure 3b the tiles are rectangles, and each tile is coloured by one of 3 colours. Depending on the algorithm, the dimensions of the tiles are $2 \times 1$ or $4 \times 2$; we refer to these tilings as the 3-*coloured* $2 \times 1$ *tiling* and the 3-*coloured* $4 \times 2$ *tiling*.

We assume that the points on the upper and left edges of a tile do not belong to the tile, so that every point in the plane belongs to exactly one tile. We write $T(p)$ for the tile that contains the point $p \in \mathbb{R}^2$ and $\chi(p) \in \{1, 2, 3, 4\}$ for the colour of the tile $T(p)$. We use the shorthand notations $T(v) = T(p(v))$ and $\chi(v) = \chi(p(v))$ for a node $v \in V$, and $T(e) = T(p(e))$ and $\chi(e) = \chi(p(e))$ for an edge $e \in E$; recall that $p(e)$ is the midpoint of $e$.

Let $v \in V$ and $e = \{u, v\} \in E$. We define the following subgraphs:

- $\mathcal{G}[v] = (V[v], E[v])$ is the connected component of $v$ in $\mathcal{G}[T(v)]$,
- $\mathcal{G}[[v]] = (V[[v]], E[[v]])$ is the connected component of $v$ in $\mathcal{G}[[T(v)]]$,
- $\mathcal{G}[e^+] = (V[e^+], E[e^+])$ is the connected component of $u$ and $v$ in $\mathcal{G}[T(e)^+]$.

See Figure 4 for an illustration.

The following lemma is used as a subroutine in the local algorithms.

**Lemma 3.** *Let $v \in V$. Assume an $a \times b$ tiling. Then (i) in time $D_{a\times b}$, node $v$ can reconstruct its own connected component $\mathcal{G}[v]$; (ii) in time $E_{a\times b}$, node $v$ can reconstruct the connected component $\mathcal{G}[[u]]$ for each $u \in B_{\mathcal{G}}(v, 1)$; (iii) in time $D_{(a+1)\times(b+1)}$, node $v$ can reconstruct the connected component $\mathcal{G}[e^+]$ for each edge $e \in E$ incident to $v$.*

*Proof.* (i) Let $r = D_{a\times b}$. In the beginning of the communication round 1, each node $v \in V$ transmits its unique identifier and the list of neighbours to each of its neighbours; the neighbours store the information they receive. In subsequent rounds, information is propagated one step further. In the end of the round $r$, each node $v$ knows the unique identifier and the list of neighbours for each node $u \in B_{\mathcal{G}}(v, r)$. Hence if $t, u \in B_{\mathcal{G}}(v, r)$, the node $v$ also knows whether $\{t, u\} \in E$, and it can reconstruct the subgraph $\mathcal{H}$ of $\mathcal{G}$ induced by $B_{\mathcal{G}}(v, r)$. Using the coordinates, the node $v$ can construct $\mathcal{H}[T(v)]$ and find its own component $\mathcal{H}[v]$.

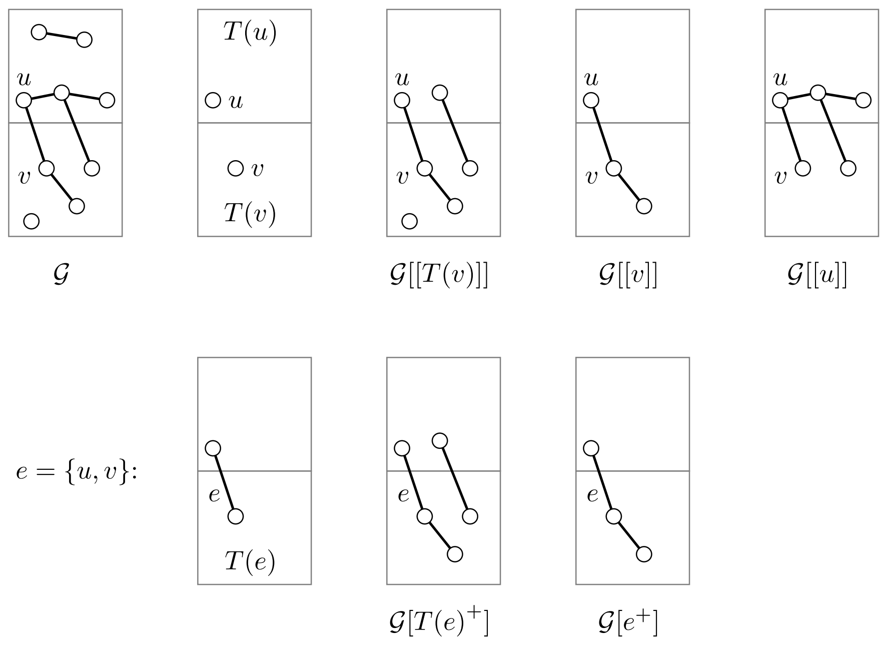

Figure 4: Connected components of induced subgraphs.

It remains to be shown that the connected component $\mathcal{H}[v]$ is equal to the connected component $\mathcal{G}[v]$. By definition, the diameter of $\mathcal{G}[v]$ is bounded by $r$. Hence $\mathcal{G}[v]$ is a subgraph of $\mathcal{H}$ and also a subgraph of $\mathcal{H}[T(v)]$.

(ii) If $\{u, v\} \in E$ and $\chi(u) \neq \chi(v)$, then $\{u, v\} \in E[[T(u)]]$ and $\{u, v\} \in E[[T(v)]]$; see Figure 4. In particular, the connected component of $u$ in $\mathcal{G}[[T(u)]]$ equals the connected component of $v$ in $\mathcal{G}[[T(u)]]$. The rest of the proof is analogous to (i).

(iii) If $\{u, v\} \in E$ then $\{u, v\} \in E[T(\{u, v\})^+]$. Since the midpoints of edges in $\mathcal{G}[T(e)^+]$ are located within an $a \times b$ rectangle, their endpoints are located within an $(a + 1) \times (b + 1)$ rectangle $B$. We can first find the connected component of $v$ in $\mathcal{G}[B]$, in time $D_{(a+1)\times(b+1)}$. Then we can discard the edges that are not in $\mathcal{G}[T(e)^+]$, and find the relevant connected component $\mathcal{G}[e^+]$. □

Note that the maximum diameter of a connected component of $\mathcal{G}[T(e)^+]$ is *not* bounded by any constant. (E.g., consider a graph consisting of arbitrarily many edges whose midpoints are contained in a tile, but whose incident vertices are not contained in the tile.) Nevertheless, $\mathcal{G}[e^+]$ can be constructed in constant time; in essence, it is possible to take shortcuts along some edges in $\mathcal{G}$ which are not in $\mathcal{G}[e^+]$.

## 3 Algorithms

In this section we present the local algorithms, and show what their local horizons are in terms of $D_{a\times b}$ or $E_{a\times b}$ (see Definitions 1 and 2). Some of the algorithms are taken from prior work, some are developed here.

Our focus is on the approximation ratio and the local horizon. We have not tried to optimise other aspects of computational complexity (e.g., total number of bits transmitted, or the amount of local computation). We describe each algorithm in a form that makes its correctness and local horizon easy to establish, and as described, the algorithms may gather some information that is only thrown away – in a real-world implementation, many shortcuts are possible.

We have divided the algorithms in three classes: the "parallel" algorithms are described in Section 3.1, the "greedy" algorithms are described in Section 3.2, and the "post-fix" algorithm is described in Section 3.3. All algorithms use the same idea of decomposing the network based on a rectangular tiling. The algorithms are presented in Tables 2–6. Before getting into the details, some general remarks are in order.

**Conventions.** Parallel algorithms in Tables 2 and 3 are straightforward: only the first step involves communication, and subsequent steps are local operations at each node. However, the greedy and post-fix algorithms in

Tables 4–6 have interleaved steps of communication and computation; we use the following conventions:

- $x(v)$, $y(v)$, and $N_v$ are local variables that are stored in the local memory of node $v$.
- All nodes execute the same algorithm synchronously in parallel. For example, in the independent set algorithm (Table 4), all nodes perform step 2a with $\chi = 4$ simultaneously in parallel.

**Consistency.** We require that all nodes execute the *same deterministic algorithm.* For example, in the algorithm for 4-approximate vertex colouring, a node $v \in V$ needs to find an optimal vertex colouring of the subgraph $\mathcal{G}[v]$. If $\mathcal{G}[u] = \mathcal{G}[v]$ for a node $u \neq v$, we assume that the node $u$ finds exactly the same colouring for $\mathcal{G}[u]$ as what $v$ finds for $\mathcal{G}[v]$. In practice, this can be achieved by using a canonical representation for $\mathcal{G}[v]$ that does not depend on $v$ but only on the sets $V[v]$ and $E[v]$; for example, we can order the nodes of $\mathcal{G}[v]$ by their unique identifiers (or coordinates), and we can order the adjacency lists by the unique identifiers of the neighbours.

An alternative way to ensure consistency of the solution to a subproblem could be to gather all information from the component at, say, lowest-identifier node, have it solve the subproblem, and propagate the solution back; this would increase the local horizon by a factor of 2 though.

**Running time.** In the model studied in this work (refer to Section 1.1 for the detailed description), local computation is free, and hence we do not pay attention to the issue of computational complexity. In particular, it does not matter which algorithm is executed at each node to produce the required output – e.g., a brute-force enumeration may be used to find an optimal colouring, or a minimum vertex cover, or any other required set. Nevertheless, it is good to note that (i) the subproblem of finding an optimal dominating set, vertex cover, or independent set within a rectangle can be solved in polynomial time [24]; (ii) for bounded-degree graphs, the size of each subproblem is bounded by a constant; and (iii) in practice, one can resort to an approximation algorithm when solving the subproblem.

## 3.1 Parallel algorithms

In the first class of algorithms, each tile solves a subproblem, *independently and in parallel*, regardless of the colour of the tile. The solutions of the subproblems are then merged trivially into the global output. The algorithms are presented in Tables 2 and 3; for the proof of correctness, see Section 4.2.

The vertex colouring algorithms and the dominating set algorithms are from prior work [23, 54, 57]; our analysis is new. The vertex cover algorithms and edge colouring algorithms resemble those from prior work [23, 54], but the tiling is different, which makes the local horizon considerably smaller (refer to Table 1).

## 3.2 Greedy algorithms

In the second class of algorithms, each tile of colour 1 solves its subproblem in a greedy manner. The solution is then greedily extended to tiles of colour 2, then to tiles of colour 3, and finally to tiles of colour 4. The algorithms are presented in Tables 4 and 5; for the proof of correctness, see Section 4.3.

The basic idea is old [3, 4], and it has been applied in the context of tile-and-combine algorithms as well [56]; our analysis is new, and the choice of the tiling for maximal matching and vertex cover is new.

## 3.3 Post-fix algorithm

The last class combines parallel and greedy algorithms. The algorithm for the maximum independent set uses a 3-coloured tiling. First, the tiles optimally solve their subproblems, independently and in parallel. This may yield a solution which is globally infeasible. The conflicts are then resolved in a greedy manner. First we resolve the conflicts that involve colour-1 nodes: we discard either conflicting colour-1 nodes, or the conflicting nodes of the other colours, whichever leaves more nodes in the independent set. Next, in a similar manner, the conflicts involving colour-2 nodes are resolved. Before the conflict resolution, the (tentative) solution has size at least that of the maximum independent set; during each of the two resolutions, at least half of the nodes is retained.

The algorithm is presented in Table 6; for the proof of correctness, see Section 4.4. The algorithm is due to Wiese [54]; our analysis is new.

**4-approximate vertex colouring**

- 4-coloured $1 \times 1$ tiling
- local horizon $D_{1\times 1}$

Algorithm for each node $v \in V$:

```
1. Using Lemma 3, find $\mathcal{G}[v]$.
2. Compute an optimal vertex colouring $C_v \colon V[v] \to \mathbb{Z}_+$ of $\mathcal{G}[v]$.
3. Output the colour $x(v) = 4C_v(v) - 4 + \chi(v)$.
```

**3-approximate vertex colouring**

- 3-coloured $2 \times 1$ tiling
- local horizon $D_{2\times 1}$

Analogous to the 4-approximation algorithm, with $x(v) = 3C_v(v) - 3 + \chi(v)$.

**4-approximate edge colouring**

- 4-coloured $1 \times 1$ tiling
- local horizon $D_{2\times 2}$

Algorithm for each node $v \in V$:

```
1. Using Lemma 3, find $\mathcal{G}[e^+]$ for edges $e$ incident to $v$.
2. For each $e \in E$ incident to $v$:
       · Compute an optimal edge colouring $C_e \colon E[e^+] \to \mathbb{Z}_+$ of $\mathcal{G}[e^+]$.
3. For each $e \in E$ incident to $v$:
       · Output the colour $x(e) = 4C_e(e) - 4 + \chi(e)$.
```

**3-approximate edge colouring**

- 3-coloured $2 \times 1$ tiling
- local horizon $D_{3\times 2}$

Analogous to the 4-approximation algorithm, with $x(e) = 3C_e(e) - 3 + \chi(e)$.

Table 2: Parallel algorithms for colouring.

**4-approximate vertex cover**

- 4-coloured $1 \times 1$ tiling
- local horizon $D_{2\times 2}$

Algorithm for each node $v \in V$:

```
1. Using Lemma 3, find $\mathcal{G}[e^+]$ for edges $e$ incident to $v$.
2. For each $e \in E$ incident to $v$:
     · Compute a minimum vertex cover $X_e$ of $\mathcal{G}[e^+]$.
3. Output "yes" if $v \in X_e$ for some $e \in E$ incident to $v$.
```

**3-approximate vertex cover**

- 3-coloured $2 \times 1$ tiling
- local horizon $D_{3\times 2}$

Analogous to the 4-approximation algorithm.

**4-approximate dominating set**

- 4-coloured $2 \times 2$ tiling
- local horizon $E_{2\times 2}$

Algorithm for each node $v \in V$:

```
1. Using Lemma 3, find $\mathcal{G}[[u]]$ for each $u \in B_{\mathcal{G}}(v, 1)$.
2. For each $u \in B_{\mathcal{G}}(v, 1)$:
     · Let $I_u = \{t \in V[[u]] : p(t) \in T(u)\}$.
     · Compute a minimum-size subset $X_u \subseteq V[[u]]$
       such that each node in $I_u \setminus X_u$ is adjacent to a node in $X_u$.
3. Output "yes" if $v \in X_u$ for some $u \in B_{\mathcal{G}}(v, 1)$.
```

$I_u$ needs to be dominated, all nodes in $V[[u]]$ can dominate.

**3-approximate dominating set**

- 3-coloured $4 \times 2$ tiling
- local horizon $E_{4\times 2}$

Analogous to the 4-approximation algorithm.

Table 3: Parallel algorithms for vertex cover and dominating set.

**Maximal independent set**

- 4-coloured $1 \times 1$ tiling
- local horizon $4D_{1\times 1} + 3$

Algorithm for each node $v \in V$:

```
1. Set $x(v) = 0$ and $y(v) = 1$.
2. For colour $\chi = 1, 2, 3, 4$:
     a. If $\chi > 1$:
          · Receive the current value of $x(u)$ from each neighbour $u$ of $v$.
          · Set $y(v) = 0$ if $x(u) = 1$ for some neighbour $u$.
     b. Using Lemma 3, find $\mathcal{G}[v]$ and $y(u)$ for all $u \in V[v]$.
     c. If $\chi = \chi(v)$:
          · Let $Y_v = \{u \in V[v] : y(u) = 1\}$.
          · Compute a maximal set $X_v \subseteq Y_v$ such that
            if $t, u \in X_v$ then $\{t, u\} \notin E[v]$.
          · If $v \in X_v$, set $x(v) = 1$.
3. Output "yes" if $x(v) = 1$.
```

For a node $v$, $x(v) = 1$ means that $v$ is already assigned to the independent set, and $y(v) = 1$ means that no neighbour of $v$ is in the independent set.

**Vertex $(\Delta + 1)$-colouring**

- 4-coloured $1 \times 1$ tiling
- local horizon $4D_{1\times 1} + 3$

Algorithm for each node $v \in V$:

```
1. Set $x(v) = \perp$ and $N_v = \emptyset$.
2. For colour $\chi = 1, 2, 3, 4$:
     a. If $\chi > 1$:
          · Receive the current value of $x(u)$ from each neighbour $u$ of $v$.
          · Set $N_v = \{x(u) : u \in B_{\mathcal{G}}(v, 1)\} \setminus \{\perp\}$.
     b. Using Lemma 3, find $\mathcal{G}[v]$ and $N_u$ for all $u \in V[v]$.
     c. If $\chi = \chi(v)$:
          · Compute a $(\Delta + 1)$-colouring $c\colon V[v] \to \mathbb{Z}_+$ of $\mathcal{G}[v]$ such that
            $c(u) \notin N_u$ for each $u \in V[v]$.
          · Set $x(v) = c(v)$.
3. Output the colour $x(v)$.
```

For a node $v$, $x(v)$ is the colour assigned to $v$ or $\perp$ if no colour is assigned yet; $N_v$ is the set of colours used by the neighbours of $v$.

Table 4: Greedy algorithms for independent set and vertex colouring.

**Maximal matching**

- 4-coloured $1 \times 1$ tiling
- local horizon $4D_{2\times 2}$

Algorithm for each node $v \in V$:

```
1. Set $x(v) = \bot$.
2. For colour $\chi = 1, 2, 3, 4$:
     a. Using Lemma 3, find $\mathcal{G}[e^+]$ and $x(t)$ for each $e \in E$ incident to $v$,
        and for each $t \in V[e^+]$.
     b. Let $F$ be the set of edges $e \in E$ incident to $v$ with $\chi(e) = \chi$. If
        $F \neq \emptyset$:
          · Choose $e \in F$.
          · Compute a maximal matching $M_e$ in $\mathcal{G}[e^+]$,
            for vertices $u$ with $x(u) = \bot$.
          · If $\{u, v\} \in M_e$ for some neighbour $u$ of $v$, set $x(v) = u$.
3. Output $x(v)$, the neighbour in the matching (or $\bot$).
```

For a node $v$, $x(v) = \bot$ if $v$ is not matched yet. When the edge $\{u, v\}$ is added to the matching, we set $x(v) = u$ and $x(u) = v$. Note that $\mathcal{G}[e^+]$ and $M_e$ do not depend on the choice of $e \in F$.

**2-approximate vertex cover**

- 4-coloured $1 \times 1$ tiling
- local horizon $4D_{2\times 2}$

Algorithm for each node $v \in V$:

```
1. Run the algorithm for maximal matching.
2. Output "yes" if $x(v) \neq \bot$.
```

Table 5: Greedy algorithms for maximal matching and vertex cover.

**4-approximate independent set**

```
· 3-coloured $4 \times 2$ tiling
· local horizon $2E_{4\times 2} + D_{4\times 2}$
```

Algorithm for each node $v \in V$:

```
1. Using Lemma 3, find $\mathcal{G}[v]$.
2. Find a maximum-size independent set $X_v \subseteq V[v]$ in $\mathcal{G}[v]$.
3. Set $x(v) = 1$ if $v \in X_v$, otherwise set $x(v) = 0$.
4. For colour $\chi = 1, 2$:
    a. Using Lemma 3, find $\mathcal{G}[[u]]$ and $x(t)$ for all $u \in B_{\mathcal{G}}(v, 1)$ and
       $t \in V[[u]]$.
    b. For each $u \in B_{\mathcal{G}}(v, 1)$ with $\chi(u) = \chi$:
        · Let $A_u = \{t \in V[[u]] : x(t) = 1, \chi(t) = \chi\}$.
        · Let $B_u = \{t \in V[[u]] : x(t) = 1, \chi(t) \neq \chi\}$.
        · If $|A_u| > |B_u|$ let $D_u = B_u$, otherwise $D_u = A_u$.
        · Set $x(v) = 0$ if $v \in D_u$.
5. Output “yes” if $x(v) = 1$.
```

Table 6: Post-fix algorithm.

## 4 Correctness of the algorithms

The following properties are used to prove the correctness of the algorithms.

**Lemma 4.** *Assume any of the tilings introduced in Section 2. Let $\{u, v\} \in E$ and $\chi(u) = \chi(v)$. Then $T(u) = T(v)$, $\mathcal{G}[T(v)] = \mathcal{G}[T(u)]$, and $\mathcal{G}[v] = \mathcal{G}[u]$.*

*Proof.* If $\chi(u) = \chi(v)$ and $T(u) \neq T(v)$ then $\|p(u) - p(v)\| > 1$ and we have $\{u, v\} \notin E$. $\square$

**Lemma 5.** *Assume any of the tilings introduced in Section 2. Let $e, f \in E$, $\chi(e) = \chi(f)$, and $T(e) \neq T(f)$. Then $V[T(e)^+]$ and $V[T(f)^+]$ are disjoint.*

*Proof.* Let $e_1 \in E[T(e)^+]$ and $e_2 \in E[T(f)^+]$. Then $\chi(e_1) = \chi(e) = \chi(f) = \chi(e_2)$ and $T(e_1) = T(e) \neq T(f) = T(e_2)$. Hence $\|p(e_1) - p(e_2)\| > 1$, and $e_1$ and $e_2$ cannot share an endpoint. $\square$

**Corollary 6.** *Assume any of the tilings introduced in Section 2. Let $e, f \in E$ and $\chi(e) = \chi(f)$. Then either $\mathcal{G}[e^+] = \mathcal{G}[f^+]$ or $V[e^+]$ and $V[f^+]$ are disjoint.*

*Proof.* If $T(e) \neq T(f)$, the claim follows from Lemma 5. Otherwise we have $\mathcal{G}[T(e)^+] = \mathcal{G}[T(f)^+]$; edges $e$ and $f$ are either in one component of $\mathcal{G}[T(e)^+]$ or in different components of $\mathcal{G}[T(f)^+]$. $\square$

**Lemma 7.** *Assume the 4-coloured $2 \times 2$ tiling or the 3-coloured $4 \times 2$ tiling. Let $u, v \in V$, $\chi(u) = \chi(v)$, and $T(u) \neq T(v)$. Then $V[[T(u)]]$ and $V[[T(v)]]$ are disjoint.*

*Proof.* Let $u_1 \in V[[T(u)]]$ and $v_1 \in V[[T(v)]]$. Then there is a node $u_2$ with $T(u_2) = T(u)$, $\chi(u_2) = \chi(u)$, and either $u_1 = u_2$ or $\{u_1, u_2\} \in E$; in both cases, $\|p(u_1) - p(u_2)\| \leq 1$. Similarly, there is a node $v_2$ with $T(v_2) = T(v)$, $\chi(v_2) = \chi(v)$, and either $v_1 = v_2$ or $\{v_1, v_2\} \in E$. Now $\chi(u_2) = \chi(u) = \chi(v) = \chi(v_2)$ and $T(u_2) = T(u) \neq T(v) = T(v_2)$. Hence $\|p(u_2) - p(v_2)\| > 2$ and $\|p(u_1) - p(v_1)\| > 0$; therefore $u_1 \neq v_1$. $\square$

**Corollary 8.** *Assume the 4-coloured $2 \times 2$ tiling or the 3-coloured $4 \times 2$ tiling. Let $u, v \in V$ and $\chi(u) = \chi(v)$. Then either $\mathcal{G}[[u]] = \mathcal{G}[[v]]$, or $V[[u]]$ and $V[[v]]$ are disjoint.*

*Proof.* Similar to Corollary 6, using Lemma 7. $\square$

### 4.1 Representatives for colours

By Corollary 6, for each colour $\chi$, there exists a set of representative edges, denoted by $R^+(\chi)$, with the following properties:

**Property 9.** *(i) $R^+(\chi) \subseteq E$. (ii) If $e \in R^+(\chi)$ then $\chi(e) = \chi$. (iii) If $e \in E$ and $\chi(e) = \chi$, then there exists a unique $x \in R^+(\chi)$, denoted by $R^+(e)$, with $\mathcal{G}[x^+] = \mathcal{G}[e^+]$. (iv) If $v \in V$, there is at most one $e \in R^+(\chi)$ such that $v \in V[e^+]$.*

By Corollary 8, for each colour $\chi$, there exists a set of representative nodes, denoted by $R(\chi)$, with the following properties:

**Property 10.** *(i) $R(\chi) \subseteq V$. (ii) If $v \in R(\chi)$ then $\chi(v) = \chi$. (iii) If $v \in V$ and $\chi(v) = \chi$, then there exists a unique $u \in R(\chi)$, denoted by $R(v)$, with $\mathcal{G}[[u]] = \mathcal{G}[[v]]$. (iv) If $v \in V$, there is at most one $u \in R(\chi)$ such that $v \in V[[u]]$.*

We emphasise that the representatives are merely used to prove the algorithms correctness; the algorithms themselves do not have to find the representatives.

## 4.2 Parallel algorithms

*Local horizon.* In each parallel algorithm, only the first step involves communication. Hence the local horizons stated in Tables 2 and 3 follow directly from Lemma 3. For example, in 4-approximate vertex colouring, each node only needs to find $\mathcal{G}[v]$ and we use a $1 \times 1$ tiling; Lemma 3(i) shows that we can complete the first step in $D_{1\times 1}$ synchronous rounds. As all other steps are local computation, the local horizon of the entire algorithm is also $D_{1\times 1}$.

**4-approximate vertex colouring.** *Feasibility.* Let $\{u, v\} \in E$ be an arbitrary edge; we need to show that $x(u) \neq x(v)$. First, assume that $\chi(u) \neq \chi(v)$, that is, $u$ and $v$ are in tiles of different colours. Then $x(u)$ and $x(v)$ differ modulo 4. Second, assume that $\chi(u) = \chi(v)$. Then by Lemma 4, $\mathcal{G}[v] = \mathcal{G}[u]$. Both $u$ and $v$ use the same deterministic algorithm to compute the colouring of the same connected component $\mathcal{G}[v]$, and thus they obtain the same colouring $C_u = C_v$ as well. In this colouring, $C_u(u) \neq C_v(v)$ and thus $x(u) \neq x(v)$.

*Approximation ratio.* If the graph $\mathcal{G}$ admits a vertex $k$-colouring, certainly the subgraph $\mathcal{G}[v]$ of $\mathcal{G}$ admits a $k$-colouring as well. Hence the largest colour that is assigned by the algorithm is $4k$.

**4-approximate edge colouring.** First we point out that the output is consistent: if $e = \{u, v\} \in E$, then both $u$ and $v$ assign the same colour $4C_e(e) - 4 + \chi(e)$ to $e$.

*Feasibility.* Let $e, f \in E$ with $e = \{t, u\}$ and $f = \{t, v\}$; we need to show that $x(e) \neq x(f)$. Clearly this is the case if $\chi(e) \neq \chi(f)$. Assume that $\chi(e) = \chi(f)$. Because $t \in V[e^+]$ and $t \in V[f^+]$, Corollary 6 implies that $\mathcal{G}[e^+] = \mathcal{G}[f^+]$. Hence the nodes $t$, $u$, and $v$ find the same colouring $C_e = C_f$, and necessarily $C_e(e) \neq C_f(f)$.

*Approximation ratio.* Similar to vertex colouring.

**4-approximate vertex cover.** Note that the sets $X_e$ constructed in the algorithm do not depend on the node $v$ which constructed it, but only on $\mathcal{G}[e^+]$.

*Feasibility.* Let $C \subseteq V$ be the set of nodes that output "yes". Consider an arbitrary edge $e = \{u, v\} \in E$; we need to show that $u \in C$ or $v \in C$. If $v \notin C$, then $v \notin X_e$. But $e \in E[e^+]$, thus $e$ needs to be covered by $X_e$, implying $u \in X_e$. Hence the node $u$ outputs "yes" and $u \in C$.

*Approximation ratio.* Let $C^*$ be an optimal vertex cover of $\mathcal{G}$. Consider one colour $\chi \in \{1, 2, 3, 4\}$. Let

$$C(\chi) = \bigcup_{e \in E : \chi(e) = \chi} X_e = \bigcup_{e \in R^+(\chi)} X_e$$

be the set of nodes chosen by the edges of colour $\chi$. Let $e \in R^+(\chi)$. The set $X'_e = C^* \cap V[e^+]$ covers each edge in $E[e^+]$. Therefore the algorithm chooses a solution $X_e$ with $|X_e| \leq |X'_e|$. By Property 9(iv), for each $v \in C^*$ there is at most one $e \in R^+(\chi)$ with $v \in V[e^+]$. Hence $|C(\chi)| \leq |C^*|$. Finally, if $v \in C$ then $v \in X_e$ for some $e \in E$; therefore $v \in C(\chi)$ for some $\chi \in \{1, 2, 3, 4\}$. Hence $C \subseteq \bigcup_\chi C(\chi)$ and $|C| \leq \sum_\chi |C(\chi)| \leq 4|C^*|$.

In essence, the algorithm constructs 4 partial vertex covers, $C(1)$, $C(2)$, $C(3)$, and $C(4)$, one for each colour. The set $C(\chi)$ covers all edges of colour $\chi$ (that is, all edges whose midpoint is within a $\chi$-coloured square). We output the union of these solutions.

The algorithm and the analysis directly generalise to the case of weighted vertex covers.

**4-approximate dominating set.** Note that the sets $I_u$ and $X_u$ constructed in the algorithm do not depend on the node $v$ which constructed them, but only on $u$.

*Feasibility.* Let $D \subseteq V$ be the set of nodes that output "yes". Consider an arbitrary $v \in V$; we need to show that if $v \notin D$, then $v$ has a neighbour $u \in D$. If $v \notin D$, then $v \notin X_v$. However, $v$ is in $T(v)$ and thus $v \in I_v$. Hence there is a node $u \in X_v$ with $\{u, v\} \in E$. Then $v \in B_{\mathcal{G}}(u, 1)$ and $u \in X_v$, and the node $u$ outputs "yes". We conclude that each node $v \notin D$ has a neighbour $u \in D$.

*Approximation ratio.* Similar to the 4-approximation algorithm for vertex cover, using Property 10. We construct 4 partial dominating sets, $D(1)$, $D(2)$, $D(3)$, and $D(4)$, one for each colour. The set $D(\chi)$ dominates all nodes of colour $\chi$; furthermore, the size of $D(\chi)$ is a lower bound for the size of $D^*$. The output is $\bigcup_\chi D(\chi)$.

In some more details, let $D^*$ be an optimal dominating set of $\mathcal{G}$. Consider one colour $\chi \in \{1, 2, 3, 4\}$. Let

$$D(\chi) = \bigcup_{v \in V : \chi(v) = \chi} X_v$$

be the nodes picked into the dominating set due to solutions computed by nodes of colour $\chi$. If $v = R(u)$ then $\mathcal{G}[[u]] = \mathcal{G}[[v]]$, $I_u = I_v$ and $X_u = X_v$. Hence

$$D(\chi) = \bigcup_{v \in R(\chi)} X_v.$$

Let $u \in R(\chi)$. If $s \in I_u$ then there is either $s \in D^*$ or there is a node $t \in D^*$ adjacent to $s$. In the latter case, $t \in V[[u]]$. Hence $X'_u = D^* \cap V[[u]]$ dominates all nodes in $I_u$. Therefore the algorithm chooses a solution $X_u$ with $|X_u| \le |X'_u|$. Furthermore, by Property 10(iv), for each $v \in D^*$ there is at most one $u \in R(\chi)$ with $v \in V[[u]]$. Hence $|D(\chi)| \le |D^*|$. Finally, if $v \in D$ then $v \in X_u$ for some $u \in V$; therefore $v \in D(\chi)$ for some $\chi \in \{1,2,3,4\}$. Hence $D \subseteq \bigcup_\chi D(\chi)$ and

$$|D| \le \sum_\chi |D(\chi)| \le 4|D^*|.$$

The algorithm and the analysis directly generalise to the case of weighted dominating sets.

## 4.3 Greedy algorithms

We only present the details for maximal independent set; other algorithms are similar.

**Maximal independent set.** *Local horizon.* To establish the local horizon of $4D_{1\times 1} + 3$, we observe that on iterations $\chi \in \{2,3,4\}$, informing the neighbours about the values $x(\cdot)$ takes 1 time unit, and on iterations $\chi \in \{1,2,3,4\}$, invoking Lemma 3 takes $D_{1\times 1}$ time units.

*Feasibility.* Let $I \subseteq V$ be the set of nodes that output "yes"; we show that $I$ is an independent set. Let $\{u,v\} \in E$ and $v \in I$. Then $x(v) = 1$ and $v$ has constructed a set $X_v$ with $v \in X_v$ on iteration $\chi(v)$. Hence $y(v)$ has remained equal to 1 on iteration $\chi(v)$. There are three cases. (i) If $\chi(u) < \chi(v)$, then $x(u) = 0$ holds before iteration $\chi(v)$, and hence it holds in the end. (ii) If $\chi(u) = \chi(v)$, then by Lemma 4, both $u$ and $v$ construct the same subsets $Y_v = Y_u$ and $X_v = X_u$. If $v \in X_v$ then $u \notin X_u$ and $x(u) = 0$ holds throughout the algorithm. (iii) If $\chi(u) > \chi(v)$, then $u$ sets $y(u) = 0$ in the beginning of iteration $\chi(v) + 1$. Hence $u \notin Y_u$ and $u \notin X_u$. In each case $u \notin I$.

*Maximality.* Assume that $v \notin I$ and $v$ does not have a neighbour in $I$. Consider the iteration $\chi(v)$. Let $U$ consist of $v$ and all neighbours of $v$ in $Y_v$. By maximality of $X_v$, we have $U \cap X_v \neq \emptyset$. Furthermore, $X_u = X_v$ for all $u \in U$. Hence at least one of the nodes $u \in U$ sets $x(u) = 1$ in the end of iteration $\chi(v)$, a contradiction.

### 4.4 Post-fix algorithm

Note that the sets $X_u$, $A_u$, $B_u$, and $D_u$ constructed in the algorithm do not depend on the node $u$ but only on its component: If $u \in V[v]$ then $X_u = X_v$. If $u \in V[[v]]$ and $\chi(u) = \chi(v) \in \{1, 2\}$, then $A_u = A_v$, $B_u = B_v$ and $D_u = D_v$.

*Local horizon.* Only steps 1 and 4a need communication. By Lemma 3, step 1 takes $D_{4\times2}$ rounds, and step 4a takes $E_{4\times2}$ rounds; step 1 is executed once and step 4a is executed twice. Hence the local horizon is $2E_{4\times2} + D_{4\times2}$.

*Feasibility.* Let $I \subseteq V$ be the set of nodes that output "yes". Let $\{u, v\} \in E$; we need to show that if $v \in I$ then $u \notin I$. First consider the case $\chi(u) = \chi(v)$. By Lemma 4, $\mathcal{G}[u] = \mathcal{G}[v]$ and $X_u = X_v$. Assume that $v \in I$; hence $v \in X_u$. Because $X_u$ is an independent set in $\mathcal{G}[u]$ and the graph $\mathcal{G}[u]$ contains the edge $\{u, v\}$, we must have $u \notin X_u$. Hence we never set $x(u) = 1$ in the algorithm, and $u \notin I$. Second consider the case $\chi(u) \neq \chi(v)$. If $v \notin X_v$ or $u \notin X_u$, the claim follows. Otherwise we have both $u \in X_u$ and $v \in X_v$. If $\chi(u) = 1$, then $u \in A_u$ and $v \in B_u$; we choose a set $D_u$ such that either $u \in D_u$ or $v \in D_u$; hence either the node $u$ sets $x(u) = 0$ after computing $D_u$, or the node $v$ sets $x(v) = 0$ after computing $D_u$; remember that $u, v \in B_{\mathcal{G}}(v, 1)$. The case $\chi(u) = 2$ is similar, and the cases $\chi(v) = 1$ and $\chi(v) = 2$ are symmetric; by assumption, we cannot have $\chi(u) = \chi(v) = 3$. Hence, when the algorithm finishes, either $x(v) = 0$ or $x(u) = 0$.

*Approximation ratio.* Let $I^*$ be a maximum-size independent set of $\mathcal{G}$. Then for each $v \in V$, the set $I^* \cap V[v]$ is an independent set of $\mathcal{G}[v]$; hence $|X_v| \geq |I^* \cap V[v]|$. Let $I_0 = \{v \in V : v \in X_v\}$ be the set of nodes that (tentatively) set $x(v) = 1$; summing over all components $\mathcal{G}[v]$ we conclude that $|I_0| \geq |I^*|$. Next we show that at least 1/2 of the set $I_0$ survives the iteration $\chi = 1$. The set $I_1$ of nodes $v \in V$ that have $x(v) = 1$ after iteration $\chi = 1$ can be written as $I_1 = I_0 \setminus D_1$ where $D_1 = \bigcup_{v \in R(1)} D_v$. Furthermore, $D_v \subseteq A_v \cup B_v = I_0 \cap V[[v]]$, the sets $V[[v]]$ are disjoint for all $v \in R(1)$, and $|D_v| \leq |A_v \cup B_v|/2$. Hence $|D_1| \leq |I_0|/2$ and $|I_1| \geq |I_0|/2$. In a similar manner, we can write the set $I$ of the nodes that survive the iteration $\chi = 2$ as $I = I_1 \setminus D_2$ with $|D_2| \leq |I_1|/2$. We conclude that $|I| \geq |I^*|/4$.

## 5 Upper bounds for $D_{a\times b}$ and $E_{a\times b}$

We proceed to derive the bounds on $D_{a\times b}$ and $E_{a\times b}$ listed in Section 1.4. Let $\mathsf{p}(x, y)$ be the maximum number of unit disks (disks of radius 1) that can be packed within an $x \times y$ rectangle; equivalently, $\mathsf{p}(x/r, y/r)$ is the maximum number of radius-$r$ disks that can be packed within an $x \times y$ rectangle.

For squares, we use the shorthand notation $\mathsf{p}(a) = \mathsf{p}(a, a)$. We now derive connections between $\mathsf{p}(x, y)$ and $D_{a\times b}, E_{a\times b}$.

## 5.1 Packing argument for $D_{a\times b}$

Fix a value for $d$ with $0 < d \leq 1$. Consider an $a \times b$ rectangle $A$ and a $d$-qUDG $\mathcal{G}$ such that the diameter of $\mathcal{G}[A]$ equals $D_{a\times b}(d)$. Let $P$ be a longest shortest path in $\mathcal{G}[A]$, that is, a shortest path between a pair of nodes that defines the diameter of $\mathcal{G}[A]$. Let $|P|$ be the number of edges in $P$. We derive an upper bound for $|P| = D_{a\times b}(d)$.

Label the nodes of $P$ with $0, 1, \ldots, |P|$. The nodes with even labels are called *black*. Because $P$ is a shortest path, there is no edge between any pair of black nodes; thus the distance between a pair of black nodes is strictly larger than $d$. We can choose an $\epsilon' > 0$ such that the distance is strictly larger than $d + \epsilon'$.

Place disks of diameter $d+\epsilon'$ centred on each black node; the disks are non-overlapping and they are contained within a rectangle of size $(a + d + \epsilon') \times (b + d + \epsilon')$. Therefore the number of black nodes is at most

$$\mathsf{p}\left(\frac{a+d+\epsilon'}{(d+\epsilon')/2}, \frac{b+d+\epsilon'}{(d+\epsilon')/2}\right) \leq \mathsf{p}\left(\frac{2a}{d} + 2 - \epsilon, \frac{2b}{d} + 2 - \epsilon\right)$$

for an $\epsilon > 0$. There are at least $(|P| + 1)/2$ black nodes; hence we conclude that there is an $\epsilon > 0$ such that

$$D_{a\times b}(d) \leq 2\mathsf{p}\big(2a/d + 2 - \epsilon,\, 2b/d + 2 - \epsilon\big) - 1. \tag{8}$$

## 5.2 Packing argument for $E_{a\times b}$

Now consider an $a \times b$ rectangle $A$ and a $d$-qUDG $\mathcal{G}$ such that the diameter of $\mathcal{G}[[A]]$ equals $E_{a\times b}(d)$. Let $P$ be a longest shortest path in $\mathcal{G}[[A]]$, let $|P|$ be the number of edges in $P$. We derive an upper bound for $|P| = E_{a\times b}(d)$.

Starting from one end of the path $P$, label the nodes $0, 1, \ldots, |P|$. For any two nodes with labels $3i$ and $3i + 1$ at least one lies within the rectangle $A$; call this node *black*. Because $P$ is a shortest path, there is no edge between any two black nodes; therefore the distance between any two black nodes is strictly larger than $d$. We can choose an $\epsilon' > 0$ such that the distance is strictly larger than $d + \epsilon'$. There are at least $|P|/3$ black nodes; hence we conclude that there is an $\epsilon > 0$ such that

$$E_{a\times b}(d) \leq 3\mathsf{p}\big(2a/d + 2 - \epsilon,\, 2b/d + 2 - \epsilon\big). \tag{9}$$

## 5.3 General case

The area of a unit disk is $\pi$. If we can pack $\mathsf{p}(x, y)$ unit disks within an $x \times y$ rectangle, then by tiling the plane with the rectangle we obtain the packing density of $\pi\mathsf{p}(x, y)/(xy)$. It is known that the densest packing of disks in the plane has density $\pi\sqrt{3}/6$ [7, 14]. Taking into account that $\mathsf{p}(x, y) \in \mathbb{N}$, we conclude that $\mathsf{p}(x, y) \leq \lfloor\sqrt{3}xy/6\rfloor$. From (8) and (9) we have that for any $0 < d \leq 1$

$$D_{a\times b}(d) \leq 2\left\lfloor \frac{2}{\sqrt{3}}\left(\frac{a}{d}+1\right)\left(\frac{b}{d}+1\right)\right\rfloor - 1, \tag{10}$$

$$E_{a\times b}(d) \leq 3\left\lfloor \frac{2}{\sqrt{3}}\left(\frac{a}{d}+1\right)\left(\frac{b}{d}+1\right)\right\rfloor. \tag{11}$$

It is not hard to see, using snake-like constructions (cf. Figure 7b), that the upper bound (10) is near-tight for small values of $d$. In what follows we focus on the case of a large $d$. Specifically, we aim at deriving tight bounds for the case of $d$ close to 1 or exactly 1, which is the case of UDGs.

## 5.4 Squares

For squares ($a = b$), we can more easily build on prior work on packings. Optimal packings of unit disks in squares are known for up to 20 disks [8, 21, 44].

A folklore result shows that 4 unit disks cannot be packed in a square smaller than $4 \times 4$, that is $\mathsf{p}(4 - \epsilon) \leq 3$ for any $\epsilon > 0$. Together with (8) this implies the upper bound $D_{1\times1}(1) \leq 5$ in (1). For the case of $1/\sqrt{2} \leq d < 1$, we can apply another folklore result $\mathsf{p}(2 + 2\sqrt{2} - \epsilon) \leq 4$ to derive the upper bound $D_{1\times1}(d) \leq 7$ in (1).

Schaer [47] shows that $\mathsf{p}(6 - \epsilon) \leq 8$; this implies the upper bound $D_{2\times2}(1) \leq 15$ in (3). For $0.843 \leq d < 1$ we can apply the upper bound $\mathsf{p}(6.747) \leq 9$ [44] to derive $D_{2\times2}(d) \leq 17$ in (3). The upper bound for $E_{2\times2}(1)$ in (6) is analogous.

## 5.5 Rectangles

Suppose that it were possible to pack 6 unit disks into a $(6 - 2\epsilon) \times (4 - 2\epsilon)$ rectangle for some $0 < \epsilon \leq 1$. Take $\epsilon' = 4 - 2\sqrt{4 - \epsilon^2} > 0$, and pack 3 disks into a $(6 - \epsilon') \times (2 + \epsilon)$ rectangle as shown in Figure 5a. Now a total of 9 disks are packed into a $(6 - \epsilon'') \times (6 - \epsilon'')$ square – a contradiction [47]. We conclude that $\mathsf{p}(6 - \epsilon, 4 - \epsilon) \leq 5$ for all $\epsilon > 0$. This implies the upper bound $D_{2\times1}(1) \leq 9$ in (2).

In an analogous way, we can derive $\mathsf{p}(8 - \epsilon, 6 - \epsilon) \leq 11$ from Wengerodt's [53] result $\mathsf{p}(8 - \epsilon) \leq 15$. This implies the upper bound $D_{3\times2}(1) \leq 21$ in (4).

Next we show that $\mathsf{p}(10 - \epsilon, 6 - \epsilon) \leq 14$. Suppose otherwise. Pack 6 disks into a $(2+\epsilon)\times(12-\epsilon')$ rectangle, 15 disks into a $(10 - 2\epsilon)\times(6 - 2\epsilon)$ rectangle, and 15 disks into another $(10 - 2\epsilon) \times (6 - 2\epsilon)$ rectangle (Figure 5b). In total, 36 disks are packed into a $(12 - \epsilon'') \times (12 - \epsilon'')$ square for a positive $\epsilon''$ – a contradiction [28]. The upper bounds $D_{4\times2}(1) \leq 27$ in (5) and $E_{4\times2}(1) \leq 42$ in (7) follow.

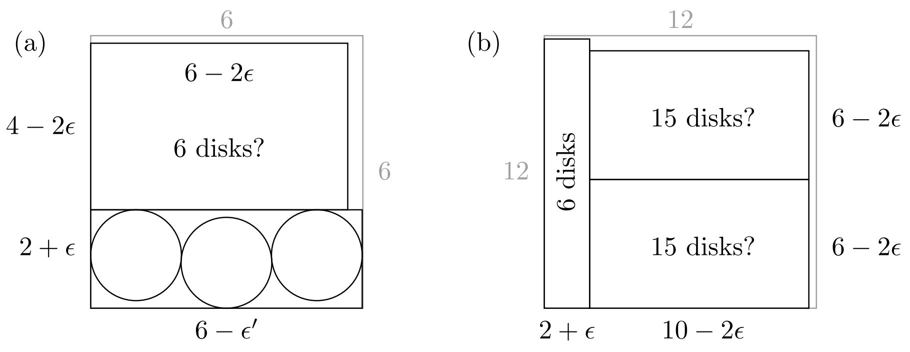

Figure 5: Packing unit disks in rectangles.

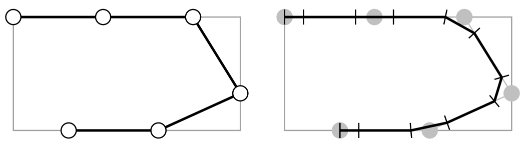

Figure 6: A general technique for obtaining lower bounds. In this illustration, $d = 1/2$ and $\epsilon = 1/6$.

Next consider a $(6.8-\epsilon) \times (4.4-\epsilon)$ rectangle. If we pack unit disks in such a rectangle, the centres of the disks are located within a $(4.8-\epsilon) \times (2.4-\epsilon)$ rectangle. Furthermore, the distance between a pair of centres is at least 2; hence each rectangle of size $(1.6-\epsilon/3) \times (1.2-\epsilon/2)$ contains at most one centre. A covering argument shows that there are at most 6 disks. Therefore $\mathsf{p}(6.8-\epsilon, 4.4-\epsilon) \leq 6$, which implies $D_{2\times 1}(d) \leq 11$ for $5/6 < d < 1$ in (2).

Finally, Peikert et al.'s [44] result $\mathsf{p}(8.532) \leq 16$ implies $\mathsf{p}(8.532, 6.532) \leq 12$. We obtain the upper bound $D_{3\times 2}(d) \leq 23$ for $0.919 \leq d < 1$ in (4).

## 6 Lower bounds for $D_{a\times b}$ and $E_{a\times b}$

Now we show that the upper bounds derived in Section 5 are tight or near-tight.

### 6.1 Constructing examples for $D_{a\times b}$

A general technique for finding tight constructions is illustrated in Figure 6. Fix $0 < d \leq 1$ and $0 < \epsilon < d/2$.

First, we find a sequence of points $P = (p_1, p_2, \ldots, p_n)$ within an $a \times b$ rectangle $A$ such that the following properties are satisfied: (i) for a pair of adjacent points $p_i, p_{i+1}$ in the sequence, the distance between $p_i$ and $p_{i+1}$ is in the range $(d + 5\epsilon/3, d + 2\epsilon]$, and (ii) for a pair of non-adjacent points $p_i, p_{i+k}$ where $k > 1$, the distance between $p_i$ and $p_{i+k}$ is larger than $d + 5\epsilon/3$.

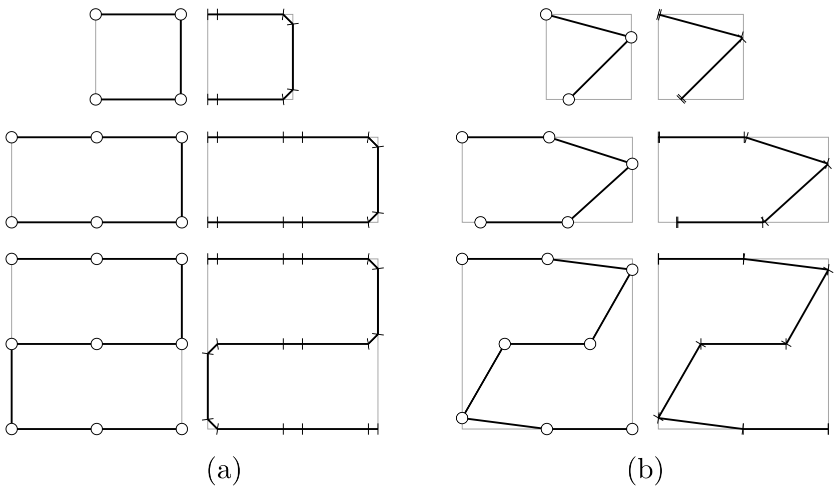
Figure 7: (a) $D_{1\times1}(d) \geq 7$, $D_{2\times1}(d) \geq 11$, and $D_{2\times2}(d) \geq 17$ for $d < 1$. (b) $D_{1\times1}(1) \geq 5$, $D_{2\times1}(1) \geq 9$, and $D_{2\times2}(1) \geq 15$.

Then, we construct a $d$-qUDG $\mathcal{G}$ as follows. The graph $\mathcal{G}$ is a path. The first node is located at $p_1$. Then, for each pair of adjacent points $p_i, p_{i+1}$ from $P$, we place two nodes along the line segment that joins $p_i$ and $p_{i+1}$, one of them $\epsilon$ units from $p_i$ and the other one $\epsilon$ units from $p_{i+1}$. Finally, the last node is located at $p_n$. In total, there are $2n$ nodes and $2n-1$ edges in $\mathcal{G}$; furthermore, $\mathcal{G} = \mathcal{G}[A]$.

### 6.2 Lower bounds for $D_{a\times b}(d)$ with $d < 1$

The lower bound constructions showing that $D_{1\times1}(d) \geq 7$, $D_{2\times1}(d) \geq 11$, and $D_{2\times2}(d) \geq 17$ for any $d < 1$ are given in Figure 7a. The left column shows the sequence of points $P = (p_1, p_2, \ldots, p_n)$; the right column shows the corresponding qUDG $\mathcal{G}$. For the sake of clarity, we present the constructions for $d = 0.8$ (and $\epsilon \approx 0.11$); they obviously generalise to any $d < 1$. The lower bounds for $D_{3\times2}(d)$ and $D_{4\times2}(d)$ are analogous. These constructions establish the lower bounds for $d < 1$ in (1)–(5).

### 6.3 Lower bounds for $D_{a\times b}(1)$

The lower bound constructions for the claims $D_{1\times1}(1) \geq 5$, $D_{2\times1}(1) \geq 9$, and $D_{2\times2}(1) \geq 15$ are given in Figure 7b. The left column shows the sequence of points $P = (p_1, p_2, \ldots, p_n)$; the right column shows the corresponding UDG $\mathcal{G}$. Recall that each point in $P$ corresponds to a pair of nodes in $\mathcal{G}$, even though the small values of $\epsilon$ makes the pair barely visible ($\epsilon < 0.002$ is sufficient for all of these). The constructions are generated and checked with a computer program. The lower bounds for $D_{3\times2}(1)$ and $D_{4\times2}(1)$ are

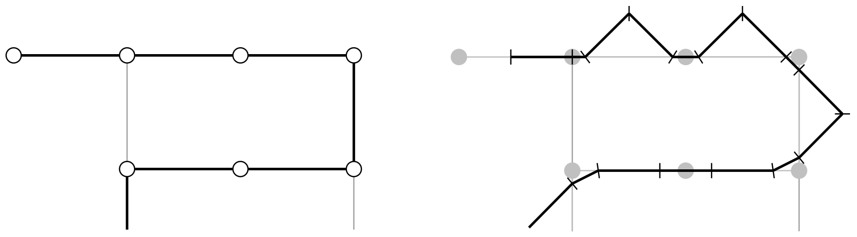

Figure 8: Obtaining lower bounds for $E_{a\times b}(d)$.

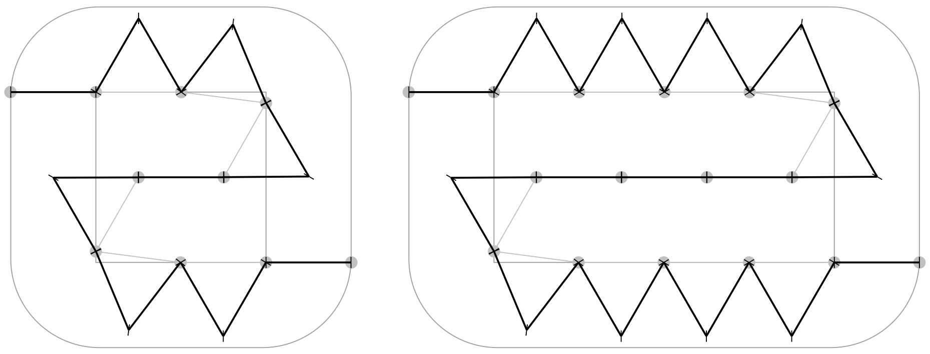

Figure 9: $E_{2\times 2}(1) \geq 23$ and $E_{4\times 2}(1) \geq 39$.

analogous. These constructions establish the lower bounds for $d = 1$ in (1)–(5).

### 6.4 Lower bounds for $E_{a\times b}(d)$

We use the same basic principle as for $D_{a\times b}$: we find a sequence of points $P = (p_1, p_2, \ldots, p_n)$. However, this time we allow for $p_1$ and $p_n$ to lie outside the $a \times b$ rectangle. Furthermore, we can replace an edge $\{p_i, p_{i+1}\}$ in $P$ with a triangle of three nodes; see Figure 8 for an illustration. Figure 9 shows the constructions for the lower bounds in (6) and (7).

## 7 Conclusions and open problems

We studied tile-and-combine local approximation algorithms for combinatorial problems in qUDGs. We gave tight bounds on the local horizons of the algorithms. The bounds are due to the connection between the horizons and maximum diameters of $d$-qUDGs induced by the nodes or edges in $a \times b$ rectangles for certain $a$ and $b$; these maximum diameters are denoted by $D_{a\times b}(d)$ and $E_{a\times b}(d)$. As functions of $d$, both $D_{a\times b}(d)$ and $E_{a\times b}(d)$ are integer-valued and non-increasing. Most interesting is the behaviour of the functions at the points of discontinuity. It follows from our results that each of the functions $D_{1\times1}(d)$, $D_{2\times1}(d)$, $D_{2\times2}(d)$, $D_{3\times2}(d)$, and $D_{4\times2}(d)$ jumps down by (at least) 2 when $d$ increases from $1-\epsilon$ to 1. What about other points: Is the value of $D_{a\times b}(d)$ always odd? Are the functions right continuous everywhere?

## Acknowledgements

We thank the anonymous reviewers for their helpful feedback. This research was supported in part by NSERC and MITACS grants, by the Academy of Finland, grants 116547, 132380, and 118653 (ALGODAN), by Helsinki Graduate School in Computer Science and Engineering (Hecse), by the Foundation of Nokia Corporation, and by the Finnish Cultural Foundation.

## References

[1] Matti Åstrand, Patrik Floréen, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. A local 2-approximation algorithm for the vertex cover problem. In *Proc. 23rd Symposium on Distributed Computing (DISC 2009)*, volume 5805 of *LNCS*, pages 191–205. Springer, 2009.

[2] Matti Åstrand and Jukka Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proc. 22nd Symposium on Parallelism in Algorithms and Architectures (SPAA 2010)*, pages 294–302. ACM Press, 2010.

[3] Hagit Attiya, Hadas Shachnai, and Tami Tamir. Local labeling and resource allocation using preprocessing. *SIAM Journal on Computing*, 28(4):1397–1413, 1999.

[4] Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *Proc. 30th Symposium on Foundations of Computer Science (FOCS 1989)*, pages 364–369. IEEE, 1989.

[5] Lali Barrière, Pierre Fraigniaud, Lata Narayanan, and Jaroslav Opatrny. Robust position-based routing in wireless ad-hoc networks with irregular transmission ranges. *Wireless Communications and Mobile Computing Journal*, 3(2):141–153, 2003.

[6] Edgar Chávez, Stefan Dobrev, Evangelos Kranakis, Jaroslav Opatrny, Ladislav Stacho, and Jorge Urrutia. Local construction of planar spanners in unit disk graphs with irregular transmission ranges. In *Proc. 7th Latin American Theoretical Informatics Symposium (LATIN 2006)*, volume 3887 of *LNCS*, pages 286–297. Springer, 2006.

[7] John H. Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups.* Springer, 1988.

[8] Hallard T. Croft, Kenneth J. Falconer, and Richard K. Guy. *Unsolved Problems in Geometry.* Springer, 1991.

[9] Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Proc. 22nd Symposium on Distributed Computing (DISC 2008)*, volume 5218 of *LNCS*, pages 78–92. Springer, 2008.

[10] Jurek Czyzowicz, Stefan Dobrev, Thomas Fevens, Hernán González-Aguilar, Evangelos Kranakis, Jaroslav Opatrny, and Jorge Urrutia. Local algorithms for dominating and connected dominating sets of unit disk graphs with location aware nodes. In *Proc. 8th Latin American Theoretical Informatics Symposium (LATIN 2008)*, volume 4957 of *LNCS*, pages 158–169. Springer, 2008.

[11] Jurek Czyzowicz, Stefan Dobrev, Hernán González-Aguilar, Rastislav Královič, Evangelos Kranakis, Jaroslav Opatrny, Ladislav Stacho, and Jorge Urrutia. Local 7-coloring for planar subgraphs of unit disk graphs. *Theoretical Computer Science*, 412(18):1696–1704, 2011.

[12] Jurek Czyzowicz, Stefan Dobrev, Evangelos Kranakis, Jaroslav Opatrny, and Jorge Urrutia. Local edge colouring of Yao-like subgraphs of unit disk graphs. *Theoretical Computer Science*, 410(14):1388–1400, 2009.

[13] Peter G. Doyle and J. Laurie Snell. *Random Walks and Electric Networks.* Number 22 in The Carus Mathematical Monographs. The Mathematical Association of America, Washington, DC, USA, 1984.

[14] László Fejes Tóth. *Lagerungen in der Ebene auf der Kugel und im Raum.* Springer, 1953.

[15] Patrik Floréen, Marja Hassinen, Joel Kaasinen, Petteri Kaski, Topi Musto, and Jukka Suomela. Local approximability of max-min and min-max linear programs. *Theory of Computing Systems*, 2010. To appear.

[16] Patrik Floréen, Marja Hassinen, Petteri Kaski, and Jukka Suomela. Tight local approximation results for max-min linear programs. In *Proc. 4th Workshop on Algorithmic Aspects of Wireless Sensor Networks (Algosensors 2008)*, volume 5389 of *LNCS*, pages 2–17. Springer, 2008.

[17] Patrik Floréen, Joel Kaasinen, Petteri Kaski, and Jukka Suomela. An optimal local approximation algorithm for max-min linear programs. In *Proc. 21st Symposium on Parallelism in Algorithms and Architectures (SPAA 2009)*, pages 260–269. ACM Press, 2009.

[18] Patrik Floréen, Petteri Kaski, Topi Musto, and Jukka Suomela. Approximating max-min linear programs with local algorithms. In *Proc. 22nd International Parallel and Distributed Processing Symposium (IPDPS 2008)*. IEEE, 2008.

[19] Patrik Floréen, Petteri Kaski, Topi Musto, and Jukka Suomela. Local approximation algorithms for scheduling problems in sensor networks. In *Proc. 3rd Workshop on Algorithmic Aspects of Wireless Sensor Networks (Algosensors 2007)*, volume 4837 of *LNCS*, pages 99–113. Springer, 2008.

[20] Patrik Floréen, Petteri Kaski, and Jukka Suomela. A distributed approximation scheme for sleep scheduling in sensor networks. In *Proc. 4th Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2007)*, pages 152–161. IEEE, 2007.

[21] Erich Friedman. Circles in squares. `http://www.stetson.edu/~efriedma/cirinsqu/`, June 2002.

[22] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, Cambridge, UK, 2005.

[23] Marja Hassinen, Valentin Polishchuk, and Jukka Suomela. Local 3-approximation algorithms for weighted dominating set and vertex cover in quasi unit-disk graphs. In *Proc. 2nd Workshop on Localized Algorithms and Protocols for Wireless Sensor Networks (LOCALGOS 2008)*, pages V.9–V.12. 2008.

[24] Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, S. S. Ravi, Daniel J. Rosenkrantz, and Richard E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *Journal of Algorithms*, 26(2):238–274, 1998.

[25] Iyad A. Kanj, Ljubomir Perković, and Ge Xia. Computing lightweight spanners locally. In *Proc. 22nd Symposium on Distributed Computing (DISC 2008)*, volume 5218 of *LNCS*, pages 365–378. Springer, 2008.

[26] Iyad A. Kanj, Andreas Wiese, and Fenghui Zhang. Local algorithms for edge colorings in UDGs. In *Proc. 35th Workshop on Graph-Theoretic Concepts in Computer Science (WG 2009)*, volume 5911 of *LNCS*, pages 202–213. Springer, 2010.

[27] Petteri Kaski, Aleksi Penttinen, and Jukka Suomela. Coordinating concurrent transmissions: A constant-factor approximation of maximum-weight independent set in local conflict graphs. *Ad Hoc & Sensor Wireless Networks: An International Journal*, 6(3–4):239–263, 2008.

[28] Kerstin Kirchner and Gerhard Wengerodt. Die dichteste Packung von 36 Kreisen in einem Quadrat. *Beiträge zur Algebra und Geometrie*, 25:147–159, 1987.

[29] Bhaskar Krishnamachari. *Networking Wireless Sensors*. Cambridge University Press, Cambridge, UK, 2005.

[30] Fabian Kuhn. *The Price of Locality: Exploring the Complexity of Distributed Coordination Primitives*. PhD thesis, ETH Zurich, 2005.

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

[32] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. On the locality of bounded growth. In *Proc. 24th Symposium on Principles of Distributed Computing (PODC 2005)*, pages 60–68. ACM Press, 2005.

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

[34] Fabian Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. *Distributed Computing*, 17(4):303–310, 2005.

[35] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Ad hoc networks beyond unit disk graphs. *Wireless Networks*, 14(5):715–729, 2008.

[36] Christoph Lenzen. *Synchronization and Symmetry Breaking in Distributed Systems*. PhD thesis, ETH Zurich, January 2011.

[37] Christoph Lenzen, Yvonne Anne Oswald, and Roger Wattenhofer. What can be approximated locally? TIK Report 331, ETH Zurich, Computer Engineering and Networks Laboratory, November 2010.

[38] Christoph Lenzen and Roger Wattenhofer. Leveraging Linial's locality limit. In *Proc. 22nd Symposium on Distributed Computing (DISC 2008)*, volume 5218 of *LNCS*, pages 394–407. Springer, 2008.

[39] Christoph Lenzen and Roger Wattenhofer. Minimum dominating set approximation in graphs of bounded arboricity. In *Proc. 24th Symposium on Distributed Computing (DISC 2010)*, volume 6343 of *LNCS*, pages 510–524. Springer, 2010.

[40] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[41] Thomas Moscibroda. *Locality, Scheduling, and Selfishness: Algorithmic Foundations of Highly Decentralized Networks*. PhD thesis, ETH Zurich, 2006.

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

[43] Christos H. Papadimitriou and Mihalis Yannakakis. Linear programming without the matrix. In *Proc. 25th Symposium on Theory of Computing (STOC 1993)*, pages 121–129. ACM Press, 1993.

[44] Ronald Peikert, Diethelm Würtz, Michael Monagan, and Claas de Groot. Packing circles in a square: A review and new results. In *Proc. 15th IFIP Conference on System Modelling and Optimization (Zürich, Switzerland, September 1991)*, volume 180 of *Lecture Notes in Control and Information Sciences*, pages 45–54. Springer, 1992.

[45] David Peleg. *Distributed Computing – A Locality-Sensitive Approach*. SIAM, 2000.

[46] Valentin Polishchuk and Jukka Suomela. A simple local 3-approximation algorithm for vertex cover. *Information Processing Letters*, 109(12):642–645, 2009.

[47] J. Schaer. The densest packing of nine circles in a square. *Canadian Mathematical Bulletin*, 8:273–277, 1965.

[48] Petra Šparl and Janez Žerovnik. 2-local 4/3-competitive algorithm for multicoloring hexagonal graphs. *Journal of Algorithms*, 55(1):29–41, 2005.

[49] Jukka Suomela. Distributed algorithms for edge dominating sets. In *Proc. 29th Symposium on Principles of Distributed Computing (PODC 2010)*, pages 365–374. ACM Press, 2010.

[50] Jukka Suomela. Survey of local algorithms. `http://www.iki.fi/jukka.suomela/local-survey`, 2011. Manuscript submitted for publication.

[51] Jorge Urrutia. Local solutions for global problems in wireless networks. *Journal of Discrete Algorithms*, 5(3):395–407, 2007.

[52] Yu Wang and Xiang-Yang Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *Mobile Networks and Applications*, 11(2):161–175, 2006.

[53] Gerhard Wengerodt. Die dichteste Packung von 16 Kreisen in einem Quadrat. *Beiträge zur Algebra und Geometrie*, 16:173–190, 1983.

[54] Andreas Wiese. Local approximation algorithms in unit disk graphs. Master's thesis, Technische Universität Berlin, 2007.

[55] Andreas Wiese and Evangelos Kranakis. Impact of locality on location aware unit disk graphs. *Algorithms*, 1:2–29, 2008.

[56] Andreas Wiese and Evangelos Kranakis. Local maximal matching and local 2-approximation for vertex cover in UDGs. In *Proc. 7th Conference on Ad-Hoc Networks & Wireless (AdHoc-NOW 2008)*, volume 5198 of *LNCS*, pages 1–14. Springer, 2008.

[57] Andreas Wiese and Evangelos Kranakis. Local construction and coloring of spanners of location aware unit disk graphs. *Discrete Mathematics, Algorithms and Applications*, 1(4):555–588, 2009.

[58] Andreas Wiese and Evangelos Kranakis. Local PTAS for dominating and connected dominating set in location aware unit disk graphs. In *Proc. 6th Workshop on Approximation and Online Algorithms (WAOA 2008)*, volume 5426 of *LNCS*, pages 227–240. Springer, 2009.

[59] Andreas Wiese and Evangelos Kranakis. Local PTAS for independent set and vertex cover in location aware unit disk graphs. *Ad Hoc & Sensor Wireless Networks: An International Journal*, 7(3–4):273–293, 2009.