Locality of Graph Problems in Directed Toroidic Grids

Kalle Viiri

Master's thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, November 2, 2020

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Kalle Viiri

Työn nimi — Arbetets titel — Title

Locality of Graph Problems in Directed Toroidic Grids

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	November 2, 2020	52

Tiivistelmä — Referat — Abstract

Locally checkable labeling problems in the LOCAL model of distributed computation are known to have only three distinct complexity classes when the attention is restricted to problems on toroidic grids only: trivial with time complexity $\Theta(1)$, local with time complexity $\Theta(\log^* n)$ and global with time complexity $\Theta(n)$. Prior work shows that problems belonging to the trivial class are easy to recognize, but that local and global labeling problems are undecidable to separate.

However, a method called algorithm synthesis exists for creating an asymptotically optimal normal-form algorithm for any locally checkable labeling problem with a time complexity of $\Theta(\log^* n)$. This process, when automated, can be used to process vast amounts of suitably encoded labeling problems in bulk, creating a more defined boundary for the undecidable class of local problems. As a proof-of-concept of this method, this work presents a new asymptotically optimal algorithm for a relaxed form of 3-coloring as well as methods for more general search of local problems.

ACM Computing Classification System (CCS):

Theory of computation—Distributed algorithms

Avainsanat — Nyckelord — Keywords
Distributed algorithms, LOCAL, Locally checkable labeling

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

Acknowledgements

I extend my sincere thanks to Jukka Suomela and Jan Studeny for the patient guidance they provided during the creation of this thesis, as well as Anni Järvenpää and the CNB study circle for their unfailing support and encouragement.

1 Introduction

In the context of theory of computation, distributed computing is a family of models of computation concerning graph problems. Unlike the more traditional models of computation such as Turing machines or random access machines, a distributed algorithm has no single state or a global memory. Instead, each node in the graph is considered to be an independent processing unit with its own state and memory, with the edges of the graph representing communication links between the nodes. The nodes begin the execution of a distributed algorithm with no information about the structure of the graph, and the algorithm must direct the nodes to communicate with each other over the edges of the graph until they have the information required to solve the problem at hand.

One interesting family of problems to study with distributed computing models is *locally checkable labelings*, or LCL [7]. A graph problem is an LCL problem if its candidate solutions consist of nodes labeled with symbols from a fixed set of output labels, and if the validity of the solution can be verified by considering only the output labels assigned to a fixed-radius neighborhood around each node separately. Common examples of LCL problems are maximal matching, maximal independent set and colouring problems. On the other hand, determining whether a path encoded in the labeling is a Hamiltonian cycle is an example of a problem that is not an LCL as observing fragments of the path around each node individually cannot guarantee that they trace a single cycle visiting each node.

The model of distributed computing used in this thesis is LOCAL [6], in which nodes have unique identifiers and communicate during synchronized communication rounds. During each communication round each node can send information to all neighboring nodes, and perform an unbounded amount of computation on any information known to it. This model lends itself well to analyzing the extent of communication required to solve a given problem. Minimizing communication between nodes is a major goal in distributed algorithm design, with the complexity of distributed algorithms in the LOCAL model being measured in terms of communication rounds required to reach a

valid solution. Some amount of communication between the nodes is often necessary for the nodes to learn about the structure of the graph or to coordinate finding a legal assignment of labels, and in the case of the hardest problems, nodes might need to pass messages across the entire graph in order to arrive to a correct solution.

A natural and simple family of graphs for studying the foundations of distributed computing is directed cycles: 2-regular graphs with a consistent orientation that allows an algorithm executing on each node to differentiate between the two neighbors of the node. Recently, research has progressed towards multi-dimensional extensions of such graphs. One such extension is the toroidic grid: a 4-regular graph with a two-dimensional orientation. The properties of LCL problems on toroidic grids are the main focus of this thesis.

Using the LOCAL model as the model of distributed computation and measuring the complexity of algorithms by communication rounds used, two important results have been established in prior work. The first of these is the speed-up lemma [1], which can be used to prove that if an LCL problem is solvable on a toroidic grid, its time complexity has to be $\Theta(1)$, $\Theta(\log^* n)$ or $\Theta(n)$ for a grid of $n \times n$ nodes. The other important result is that while LCL problems with a time complexity of $\Theta(1)$ are easily recognized, determining the time complexity of an LCL between $\Theta(n)$ and $\Theta(\log^* n)$ is undecidable in toroidic grids [1]. Both of these results will be explained in greater detail in this thesis.

Despite classifying an LCL problem's time complexity on a grid being an undecidable problem in the general case, a process by Brandt et al. [1] called *algorithm synthesis* exists for recognizing problems of complexity $O(\log^* n)$. Given the specification for an LCL problem, the process constructs an asymptotically optimal algorithm in finite time unless the problem's complexity is $\Omega(n)$. The method works by reducing the problem to the computation of a maximal independent set, known to be computable in $O(\log^* n)$ communication rounds [9].

Algorithm synthesis requires an unbounded amount of computation, and will never succeed for problems of complexity $\Omega(n)$. Regardless, the process allows one to recognize many LCL problems of complexity $\Theta(\log^* n)$ among the

set of all possible problems. The aim of this thesis is to present the theoretical background of algorithm synthesis, implement the process and show results of LCL problems proven to be solvable in $O(\log^* n)$ communication rounds.

2 Prior work

The LOCAL model of distributed computation used in this work was introduced by Linial [6]. While many individual LCL problems had been studied prior, Naor and Stockmeyer[7] defined the concept of LCL problems and presented weak 2-coloring as the first concrete example of a non-trivial LCL solvable by a constant-time algorithm in the LOCAL model. They also showed that if a given LCL problem can be solved by a constant-time randomized algorithm, it also can be solved by a constant-time deterministic algorithm.

One particularly important result by Naor and Stockmeyer[7] is the order-invariance theorem, which states that constraining distributed algorithms to refer only to the relative values of node identifiers instead of absolute values does not reduce the computational capabilities of the LOCAL model. A proof for this theorem is included in this thesis, in section 4.4.

Chang and Pettie [2] expanded upon the order-invariance theorem to prove the existence of a complexity gap that no LCL problem requires between $\omega(1)$ and $o(\log^* n)$ communication rounds to compute. A theorem by Jukka Suomela presented by Chang and Pettie [2] furthermore proves that all LCL problems with time complexity of $O(1)$ are trivial in toroidic grids in the sense that they permit solutions where each node outputs the same, constant output label. A proof for this theorem is presented in Section 5.1.

Brandt et al. [1] present a proof that no LCL problem has a time complexity between $\omega(\log^* n)$ and $o(n)$ in toroidic grids. This proof is included in this thesis in section 5.3. They also show that determining a LCL problem's time complexity between $\Theta(\log^* n)$ and $\Theta(n)$ is undecidable in toroidic grids. However, they also present algorithm synthesis, a method for automatically finding asymptotically optimal algorithms for problems with time complexity $O(\log^* n)$. This method was implemented and applied as a part of this thesis, with the theory of synthesis explained in Section 6 and notes on practical use

to categorize problems by complexity in Section 7.

3 Definitions

A graph is a pair (V, E) where V is a finite set of nodes and $E \subseteq (V \times V)$ is the set of edges of the graph. E is a symmetric relation, and when $e = (u, v) \in E$ we consider u and v to be connected by the edge e. For any edge $e = (u, v)$, the inverse edge (v, u) can be denoted as e^R. No edges are allowed to connect a node to itself, so $(u, u) \notin E$ for all $u \in V$ unless otherwise specified. The set of nodes of a graph G is denoted as V_G and the set of edges as E_G.

Two nodes $u, v \in V$ are considered to be *neighbors* when $(u, v) \in E$. The number of distinct neighbors in a graph G for $u \in V_G$ is called the degree of u, and denoted as $\deg(u)$.

A graph is *n-regular* when $\deg(u) = n$ for all $u \in V$. For example a graph where each node has four neighbors can be referred to as being 4-regular.

In many applications of graphs in distributed computing it is desirable to consider edges as having additional information such as a port number or a direction. For this purpose, an *edge-labeled graph* is defined as a tuple (V, E, ℓ) where (V, E) is a graph and $\ell : E \to L$ is a function that assigns each edge an edge label from a finite set of legal labels L. The edge labeling does not have to be commutative: for an edge $e \in E$, it is possible for $\ell(e) \neq \ell(e^R)$.

The edge-labeled graph is a generalization of normal graphs: every graph $G = (V, E)$ can be represented as an edge-labeled graph (V, E, ℓ) by choosing L with $|L| = 1$. Since this thesis primarily concerns edge-labeled graphs, the term *graph* will be used to refer to edge-labeled graphs from here on. The edge labeling of a graph G is denoted as ℓ_G, and when omitted for a graph G, ℓ_G can be assumed to label every edge with the same constant "null" label.

A graph is is *connected* when for every distinct $u, w \in V$ there exists an ordered sequence of nodes $(v_1, v_2, \ldots, v_n) \subset V$ such that

$$\{(u, v_1), (v_1, v_2), \ldots, (v_n, w)\} \subseteq E$$

.This set of edges is called a *path* between u and w and the number of elements

in the set is the length of the path. In the scope of this thesis, graphs are assumed to be connected unless stated otherwise.

Two graphs with an identical structure are called *isomorphic*. More formally, graphs G and H are isomorphic when a bijection $f : V_G \to V_H$ exists such that $(u, v) \in E_G$ if and only if $(f(u), f(v)) \in E_H$. Such f is called an isomorphism of G and H. Within the context of this thesis, f is also required to satisfy the condition that $\ell_G(u, v) = \ell_H(f(u), f(v))$ for all $(u, v) \in E_G$ and $(f(u), f(v)) \in E_H$.

The *distance* between two nodes in a graph is the minimum number of edges that form a path between the nodes. For a graph G, the distance between the nodes $u, v \in V_G$ is denoted as $\text{dist}_G(u, v)$. The distance between a node u and an edge e is similarly the minimum number of edges that form a path from u through e. When $u \in V_G$ and $e = (v, w) \in E_G$ let $\text{dist}_G(u, e)$ be defined as $\min\{\text{dist}_G(u, v), \text{dist}_G(u, w)\} + 1$.

A graph H is a *subgraph* of G if $V_H \subseteq V_G$, $E_H \subseteq E_G$ and $\ell_H(e) = \ell_G(e)$ for every $e \in E_H$. When $W \subset V_G$, F is an *induced subgraph* of G when $V_F = W$ and $E_F = \{(u, v) \in E_G : u \in W \wedge v \in W\}$. For a graph G and $W \subset V_G$, the subgraph induced by W is denoted as $G[W]$.

In the context of locality, it is usually convenient to restrict our attention to limited-sized subgraphs centered on a particular node. For $u \in V_G$ and some $r \in \mathbb{N}$, the *r-radius neighborhood* of u is $G[V']$ where

$$V' = \{v \in V_G : \text{dist}(u, v) \leq r\}$$

. For a radius r, graph G and node $u \in V_G$, the r-radius neighborhood around u is denoted as $B_G(u, r)$. As it is sometimes useful to consider the neighborhoods of large sets of nodes at a time, for any $S \subseteq V_G$, the r-radius neighborhood of S is defined as

$$B_G(S, r) = \bigcup_{u \in S} B_G(u, r)$$

.

A *centered graph* of radius r is a pair (H, u) where H is a graph and

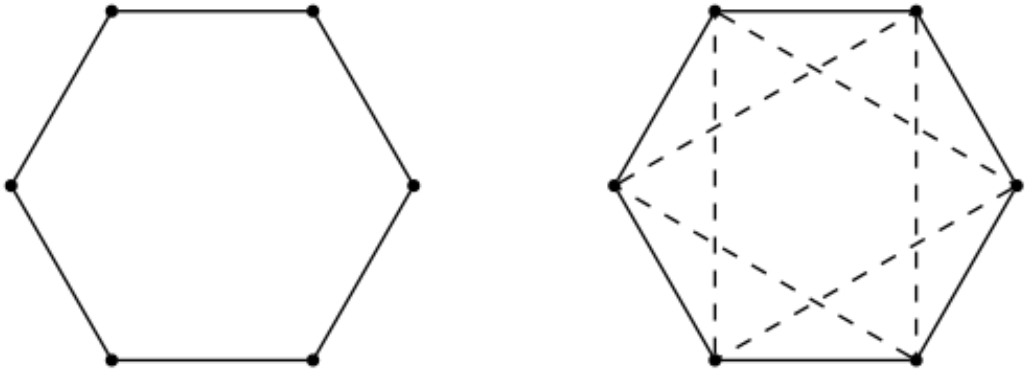

Figure 1: On the left there is a six-noded graph G. Pictured to its right is G^2 with the newly added edges denoted by dashed lines.

$u \in V_H$, where $\text{dist}(u, v) <= r$ and $\text{dist}(u, e) <= r$ for all $v \in V_H$ and $e \in E_H$.

The nth *power graph* of a graph G is a graph with added edges connecting nodes at most n edges away. For a more precise definition, $G^n = (V_G, E^n)$ where $E^n = \{(u, v) \in V \times V : u \neq v \wedge \text{dist}_G(u, v) \leq n\}$. An example of a power graph is pictured in Figure 1.

The *iterated logarithm* of n, denoted as $\log^* n$, is the least integer j for which $\log^j n \geq 1$.

3.1 Cycles and grids

A graph is called a *cycle* when it is connected and 2-regular. A cycle G is a *directed cycle* if ℓ_G defines a consistent orientation for G. In order to be considered a consistent orientation, ℓ_G must satisfy the following properties: $\ell_G(e) \neq \ell_G(e^R)$ for every $e \in E_G$ and $\ell_G(u, v) \neq \ell(u, w)$ for any distinct $(u, v), (u, w) \in E_G$. In addition, L must consist of exactly two labels. This creates two "directions" in the graph, with each node having a single "predecessor" and a single "successor". By traveling from a node to its successor and continuing recursively, one traces a path through all the nodes in the graph. Traveling from the same node through predecessors recursively creates the same path in reverse.

A *4-regular toroidic grid*, referred to as *grid* from now on, is a 4-regular connected graph that is formed by extending a cycle into a second dimension

as follows. Let w and h be positive integers, and $w \times h$ the *dimensions* of the grid G. Then $|V_G| = wh$. Each node u is associated with a unique coordinate pair from $\{0, 1, \ldots, w-1\} \times \{0, 1, \ldots, h-1\}$. The components of the coordinate pair of a node u will be denoted as x_u and y_u.

Let the edges of the grid be defined as follows. The group of edges traversing the y component will be denoted as N and S such that

$$\mathsf{N} = \{(u, v) \in V_G^2 : y_u + 1 \equiv y_v \pmod{h}\}$$

and $\mathsf{S} = \mathsf{N}^{-1}$. Similarly, E and W shall denote the sets of edges traversing the x component such that

$$\mathsf{E} = \{(u, v) \in V_G^2 : x_u + 1 \equiv x_v \pmod{w}\}$$

and $\mathsf{W} = \mathsf{E}^{-1}$. The union of these sets forms the edges of the grid with $E_G = \mathsf{N} \cup \mathsf{E} \cup \mathsf{S} \cup \mathsf{W}$.

Like cycles, grids can be directed. A *directed grid* has an edge labeling that defines two consistent orientations, one on each axis. These orientations shall be referred to with the compass directions north, east, south and west, denoted as $\{\mathcal{N}, \mathcal{E}, \mathcal{S}, \mathcal{W}\}$ respectively.

More formally, a grid G is a directed grid if the edge labeling $\ell_G : E_G \to L$ with $L = \{\mathcal{N}, \mathcal{E}, \mathcal{S}, \mathcal{W}\}$ labels the edges as follows. For each $e \in E_G$,

$$\ell_G(e) = \begin{cases} \mathcal{N} & \text{if } e \in \mathsf{N} \\ \mathcal{E} & \text{if } e \in \mathsf{E} \\ \mathcal{S} & \text{if } e \in \mathsf{S} \\ \mathcal{W} & \text{if } e \in \mathsf{W} \end{cases}$$

Connected induced subgraphs of grids can be referred to as *subgrids* in future discussion.

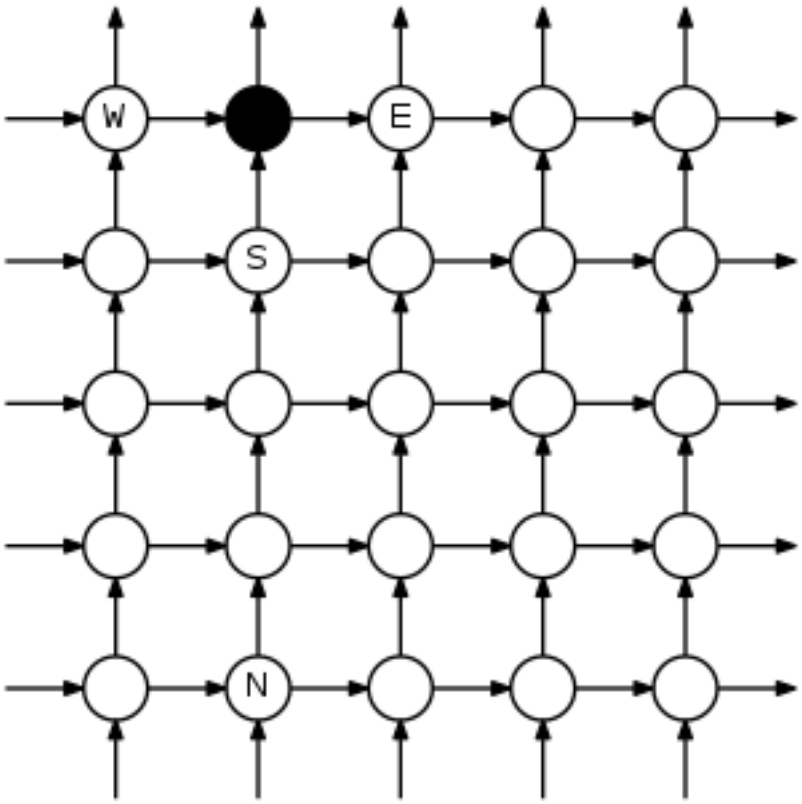

Figure 2: Pictured above is a grid with dimensions 5×5. The neighbors of the black node have been marked $\mathsf{N}, \mathsf{E}, \mathsf{S}$ ja W corresponding to northern, eastern, southern and western neighbor, respectively. The direction of the arrows indicates northbound and eastbound edges. The grid has no boundary, but rather it wrapping around from north to south and east to west creates a toroidic structure.

4 Distributed algorithms

Distributed computing is a field of computational theory focusing on graphs that model networks of computers. The nodes of the graph represent independent processing units, with the edges representing connections between them. The nodes are generally not aware of the entirety of the graph they are in, but have to expand their knowledge of the structure of the graph by exchanging knowledge between their neighbors similarly to how real life peer-to-peer networks function.

There are many different models and approaches to study distributed computation. In this work, the LOCAL model by Linial [6] is used. In the LOCAL model, each node is assumed to follow the same deterministic algorithm, with communication progressing in synchronized communication cycles. During each cycle, each node can send each of its neighbors an arbitrarily long message. After a cycle and before the next cycle is started, each node can perform an arbitrarily complex computation using any information it has access to.

When studying a LOCAL algorithm for the graph G, each node is considered to have a unique integer identifier. The unique identifier is given as an injective function $\mathrm{id} : V_G \to S$ where S is $\{1, 2, \ldots, |V_G|^k\}$ for a constant k. Notice this means each identifier in S is therefore limited to a maximum length of $O(\log n)$ bits unless specified otherwise. When the algorithm is executed, each node knows initially only its own identifier, and can learn the identifiers of the other nodes by exchanging this information with its neighbors. An algorithm is considered correct only if it provides a correct output regardless of the identifier assignment, so the assignment can be considered to be adversarial.

Each node executing a distributed algorithm can halt by printing an output label from a finite set of eligible labels denoted as Γ. The algorithm is considered to have halted when each node has printed an output. The outputs of the nodes are interpreted as the solution to the problem. For example, when solving the k-coloring problem, $\Gamma = \{1, 2, \ldots, k\}$ and the outputs must be arranged so that no two neighboring nodes output the same value.

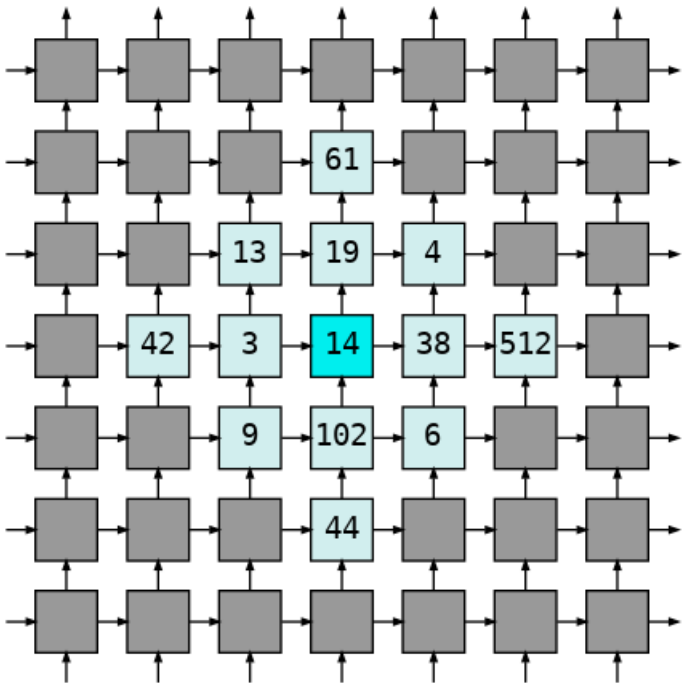

Figure 3: After two communication rounds, the node with the identifier 14 knows the identifiers and inputs of the indicated nodes.

Sometimes it is desirable that the nodes of the graph receive input. The role of the input can vary: it can introduce new constraints to the problem or provide advice that reduces the complexity of the problem in order to study the effects of extra information on the communication time required. The set of eligible input symbols is denoted as Σ. Each node solving a problem on graph G is assumed to have access to a value specified by the function $\text{input} : V_G \to \Sigma$ that defines the input for each node of the graph. If input is not mentioned for a given problem, Σ can be assumed to consist of a single null symbol that does not convey meaningful information pertaining to the problem or G. In the beginning of an algorithm's execution for G, each node $u \in V_G$ knows only the values $\text{id}(u)$ and $\text{input}(u)$, the edges connecting them to other nodes, and the edge labels for these edges. Importantly, in the case

of directed grids, the node doesn't know its co-ordinates within the grid or the dimensions of the grid unless these are provided as input.

Since there is no restriction on the amount of data a node can send during a communication round, one can assume each node sends all of their information each round to each of their neighbors. This results in each node u knowing the identifiers and inputs of all nodes in $B_G(u, n)$ after n rounds of communication. An example is seen in Figure 3.

This thesis only discusses distributed algorithms that are deterministic. Without randomness, an algorithm A that executes in t communication rounds on a graph G will assign the output value to a node $u \in V_G$ depending only on the information available within $B_G(u, t)$. This means that a deterministic algorithm can be thought of a function whose domain is a set of tuples $(K, u, \mathrm{id}_K, \mathrm{input}_K)$, where (K, u) is a centered graph, $\mathrm{id}_K : V_K \to S$ is a function assigning each node a unique identifier from S and $\mathrm{input}_K : V_K \to \Sigma$ is the input assignment. Each such tuple maps to a symbol $\gamma \in \Gamma$ which is the output label A assigns to u.

4.1 Time complexity

Recall that in the LOCAL model of distributed computing, the time complexity of an algorithm is measured in the communication rounds required to produce the desired output. In the end of each communication cycle, each node can perform an arbitrary amount of computation using any information it has received. This computation is not taken into account when determining an algorithm's time complexity, nor is the length of the messages sent or received during each communication round.

If every node knows the structure of the entire graph, including the edge labels, inputs and the identifiers, the problem can be solved immediately within each node without any further communication if a correct solution exists. Each node can then choose its output to conform to this solution. This means that assuming a solution exists to a given graph problem, there exists a distributed algorithm that finds the solution by propagating information within the graph until each node has complete knowledge of the graph's

layout and can solve the problem locally. Therefore, for a graph with n nodes, $O(n)$ is the absolute upper bound for time complexity in the LOCAL model. Problems that require $\Omega(n)$ communication rounds are called *global* problems.

Many graph problems are solvable only for some graphs. For example, 2-coloring is only possible for bipartite graphs, or in the case of grids, those with even dimensions. Problems that are unsolvable for infinitely many graphs are considered to have a time complexity of $\Omega(n)$ and to be inherently global.

Other typical time complexities for graph problems in LOCAL are constant time problems solvable in $O(1)$ communication rounds, and the iterated logarithm time class $O(\log^* n)$. Due to $\log^* n$ being a very slow-growing function, problems whose time complexity is $O(\log^* n)$ will be referred to as *local* problems.

4.2 Locally checkable labelings

In addition to the time complexity required to solve a particular problem, it is often of interest whether candidate solutions for a particular problem can be determined by observing only the surroundings of each node. Vertex coloring problems are a common example: the correctness of a given coloring for graph G can be ensured by only checking the outputs $B_G(u, 1)$ for each $u \in V_G$, and ensuring no node in the neighborhood outputs the same label as u.

Problems like this, where the problem requires each vertex to receive a label from a fixed set of input labels and whose candidate solutions can be verified in $O(1)$ communication rounds, are called *locally checkable labelings*, abbreviated from here on as LCL. This thesis uses the definition of LCL by Naor and Stockmeyer [7] extended with edge labeling. An LCL problem $\mathcal{L}$ consists of a *checkability radius* $r \in \mathbb{Z}_+$, a finite set of input symbols Σ, a finite set of output symbols Γ and a finite set of *locally consistent labelings* $\mathcal{C}$. Each element of $\mathcal{C}$ is a tuple $(H, u, \text{input}_H, \lambda_H)$, where (H, u) is a centered graph for which $\text{dist}_H(u, v) \leq r$ for all $v \in V_H$, and $\text{input}_H : V_H \to \Sigma$ and $\lambda_H : V_H \to \Gamma$ are functions specifying the input and output labelings for V_H, respectively.

Given a graph G and a function input : $V_G \to \Sigma$ specifying input labeling, an output labeling $\lambda : V_G \to \Gamma$ is $\mathcal{L}$-*legal* if for every $u \in V_G$ there exists a $c = (H, v, \text{input}_H, \lambda_H) \in \mathcal{C}$ and an isomorphism $\pi : B_G(u, r) \to H$ such that $\pi(u) = v$ and input, output and edge labelings are preserved. If there exists even a single $u \in V$ whose r-radius neighborhood cannot be mapped to one of the eligible neighborhood labelings in $\mathcal{C}$ by such isomorphism, the λ is not a $\mathcal{L}$-legal labeling.

4.3 Checkability radius

Every locally consistent labeling for an LCL problem is a centered subgraph of a radius bounded by a problem-specific constant r. This constant is called the checkability radius of $\mathcal{L}$ because it specifies the maximum distance from each node that must be observed in order to ascertain the legality of an output labeling.

When $\mathcal{L}$ is an LCL with a checkability radius of r, it is possible to create a new LCL $\mathcal{L}'$ with a checkability radius of 1 such that any output labeling λ for $\mathcal{L}$ can be converted to an output labeling λ' for $\mathcal{L}'$ in $O(r)$ communication rounds, with λ' being a $\mathcal{L}'$-legal labeling if and only if λ is a $\mathcal{L}$-legal labeling. This is accomplished by defining the output labels of $\mathcal{L}'$ such that each output label of $\mathcal{L}'$ corresponds to a r-radius eligible neighborhood labeling of $\mathcal{L}$.

Assuming an algorithm A that solves $\mathcal{L}$, it can be executed with an alteration to solve $\mathcal{L}'$: instead of printing outputs, each node u spends r communication rounds to collect all output labels in $B_G(u, r)$ and sets its label . In this manner, each node knows the outputs for $\mathcal{L}$ and produce a legal labeling for $\mathcal{L}'$.

Since any LCL can be converted in this fashion to a problem with a checking radius of 1, the checking radius can be assumed to be 1 without loss of generality. Since the above transformation only requires r extra computation steps and r is a constant that only depends on the original LCL, the extra communication required is $O(1)$ and can be ignored in time complexity analysis.

4.4 Order-invariance

An *order-invariant algorithm* is a distributed algorithm satisfying an additional constraint for computation requiring that the algorithm does not refer to the concrete values of the identifiers assigned to each node, but only their relative ordering [7]. More formally, two identifier assignments $\eta, \eta' : V \to \mathbb{Z}_+$ are *order-equivalent* for a given set of nodes V if for every $u, v \in V$, $\eta(u) < \eta(v)$ if and only if $\eta'(u) < \eta'(v)$. A distributed algorithm A is then order-invariant if for every input-labeled centered graph $(K, u, \eta, \text{input}_K)$ in the domain of A, if η and η' are order-equivalent then $A(K, u, \eta, \text{input}) = A(K, u, \eta', \text{input})$.

Naor and Stockmeyer established that if an algorithm A solves an LCL $\mathcal{L}$ with a constant time bound $t = O(1)$, it follows that there exists an order-invariant distributed algorithm A' that solves $\mathcal{L}$ with a time bound t [7]. The importance of the proof is considerable not only because of order-invariance itself, but because it can be extended to prove gaps in the complexity space for LCL problems.

A concept required by this and further proofs is the *multicolor hypergraph Ramsey number*, referred to just Ramsey number for short. For a set S and positive integer $p \leq |S|$, let $[S]^p$ denote the set $\{A \subseteq S : |A| = p\}$. Each element of $[S]^p$ can be conceptualized as a *hyperedge* connecting p elements together in a hypergraph similarly to how an edge connects two elements in a normal graph. The multicolor hypergraph Ramsey number is defined by the following theorem [5]:

Lemma (Multicolor hypergraph Ramsey theorem)**.** *For any p, m and c, there is a positive integer $R(p, m, c)$ only depending on the values of p, m and c such that the following holds. Let S be a set with $|S| > R(p, m, c)$, and with each element of $[S]^p$ assigned a single color from a set of c distinct colors. For any way colors are assigned, there exists a $T \subseteq S$ with $|T| = m$ such that each element of $[T]^p$ is assigned the same color.*

The order-invariance theorem by Naor and Stockmeyer [7] follows.

Theorem 1. *Fix a LCL $\mathcal{L}$, a class of graphs $\mathcal{G}$ and a constant integer time bound t. Let d be the maximum degree among all the graphs of $\mathcal{G}$. Then*

there is a number R depending only on $\mathcal{L}$, t and d such that the following holds. For every algorithm A that executes in t or fewer communication steps and every set of identifiers S with $|S| > R$, there is an order-invariant local algorithm A' that, for every $G \in \mathcal{G}$ and every input labeling $\text{input}_G : V_G \to \Sigma$, if A outputs a $\mathcal{L}$-legal labeling for G for any identifier assignment, then A' outputs a $\mathcal{L}$-legal labeling for G for any identifier assignment.

Proof. By assumption, A is an algorithm that can label any $G \in \mathcal{G}$ correctly when the node identifiers are drawn from S. As A completes in at most t communication rounds, it can be considered a finite function from t-radius input-labeled, id-numbered centered graphs to the output label set of $\mathcal{L}$. Let

$$\{(K_1, s_1, \text{input}_1), (K_2, s_2, \text{input}_2), \ldots, (K_z, s_z, \text{input}_z)\}$$

be the set of distinct input-labeled centered graphs $(K_i, s_i, \text{input}_i)$ of radius t such that given an identifier function $\eta_i : V_{K_i} \to S$ for a set of identifiers S, $(K_i, s_i, \eta_i, \text{input}_i)$ is in the domain of A. The maximum number of nodes in all K_i will be denoted as p. The values of p and z depend only on the choice of $\mathcal{L}$, t, as well as the the maximum degree of graphs in $\mathcal{G}$ which will be denoted as d.

Let S be any set of possible identifiers and $X, X' \in [S]^p$ such that $X = \{x_1, x_2, \ldots, x_p\}$ and $X' = \{x'_1, x'_2, \ldots, x'_p\}$ with both indexed in an increasing order by the value of the identifier. Considering the node sets of $K_1, K_2, \ldots, K_z$ as being disjoint of one another, let $V = \bigcup_{i=1}^{z} V_{K_i}$.

Consider any function $\sigma : V \to \{1, 2, \ldots, p\}$ with the additional restriction that $\sigma(u) \neq \sigma(v)$ for all distinct $u, v \in V_{K_j}$ for each j. Let $K_j(\sigma)$ be the input-labeled and id-labeled centered graph $(K_j, s_j, \text{id}, \text{input}_j)$ with $\text{id}(u) = x_{\sigma(u)}$ for all $u \in V_{K_j}$. Similarly, let $K'_j(\sigma)$ be the input-labeled and id-labeled centered graph $(K_j, s_j, \text{id}', \text{input}_j)$ with $\text{id}'(u) = x_{\sigma(u)}$. Note that id and id$'$ are order-equivalent id assignments.

Using σ, let $X \equiv X'$ iff A produces the same output for order-equivalent identifier assignments from identifier sets X and X', in other words when $A(K_j(\sigma)) = A(K'_j(\sigma))$ for all σ and j. This is an equivalence relation. An upper bound c exists on the number of equivalence classes, depending only

on p, z and $\mathcal{L}$, and therefore by extension, only d, t and $\mathcal{L}$.

Let r be the checking radius of $\mathcal{L}$. Let m equal p plus the maximum number of nodes in any $(r+t)-$radius centered graph with degree of at most d. Now m depends only on $\mathcal{L}$, t and d.

Let $R = R(p, m, c)$. Following from Ramsey theory, R depends only on the choice of p, m and c and therefore only on $\mathcal{L}$, t and d. Therefore, treating the equivalence classes of subsets of S as colors, the following is true. For any choice of identifier set S with $|S| \geq R$ and any assignment of $[S]^p$ to the c distinct equivalence classes, there exists $T \subseteq S$ with $|T| = m$ such that all of $[T]^p$ is assigned the same equivalence class.

Let U equal T with the p greatest identifiers removed, leaving $|U|$ equal to the maximum number of nodes in any centered graph with a maximum degree of d and a radius of at most $r+t$. Let $(H, s, \eta, \text{input})$ and $(H, s, \eta', \text{input})$ be any input-labeled centered graphs in the domain of A with η, η' being any identifier assignments $\eta, \eta' : V_H \to U$.

Let $X, X' \in [T]^p$ be any sets containing all identifiers assigned by η and η' to V_H respectively with the additional constraint that for every $u \in V_H$, if $\eta(u) = x_i$ then $\eta'(u) = x'_i$ retaining the indexing in increasing order as before. For cases where $|V_H| < p$, η and η' can be "padded" to use elements of $T \setminus U$ to still allow $|X| = |X'| = p$, as the p greatest identifiers in T are not present in U.

Let σ and j be chosen such that $(H, s, \eta, \text{input}) = (K_j(\sigma))$. It follows that $(H, s, \eta', \text{input}) = (K'_j(\sigma))$. As $X \equiv X'$ it follows that $A(H, s, \eta, \text{input}) = A(H, s, \eta', \text{input})$. This means A is order-invariant when U is used as the set of identifiers.

The order-invariant algorithm A' can be derived from A as follows. On a centered id-labeled graph $(H, u, \text{id}, \text{input})$, first A' creates a new identifier assignment $\text{id}' : V_H \to U$ such that id and id' are order-equivalent for V_H. Then A' labels u with the label A would choose for $(H, u, \text{id}', \text{input})$. As any two order-equivalent assignments of identifiers result in an order-equivalent identifier set drawn from U, clearly A' is order-invariant. $\square$

5 Time complexity classification in grids

If the graph is arbitrarily chosen, LCL problems can belong to a wide variety of different time complexity classes. However, when resricting the graph family to directed grids, every LCL problem that takes no input belongs to one of three time complexity classes [1][2]. These classes are $\Theta(1)$, $\Theta(\log^* n)$ and $\Theta(n)$. Problems belonging to these classes will be referred to as *trivial*, *local* and *global* problems, respectively.

The global problems consist furthermore of three distinct families of problems. Some LCL problems are solvable in all sufficiently large grids. For example, a legal 3-coloring always exists for a directed grid of dimensions $n \times n$ if $n > 1$. Other LCL problems are global because they are unsolvable for infinitely many graphs, for example 2-coloring which is only solvable when n is even. Brandt et al.[1] note that such problems are global in $n \times n$ grids even if the algorithm is allowed to assume a value for n for which a solution exists. Finally, the LCL problem can just be ill-defined in a way that prohibits a solution from existing in the first place. Regardless of the particular reason, problems that cannot be solved by an algorithm of time complexity $o(n)$ will be considered global within the context of this thesis.

5.1 Constant time problems

The simplest complexity class of LCL problems to recognize in toroidic grids is $O(1)$. These problems can be solved in a constant number of communication rounds, no matter the identifier assignment or the size of the grid. In general grids, these problems include non-trivial problems such as weak 2-coloring as proven by Naor and Stockmeyer [7].

In toroidic grids, however, every LCL problem $\mathcal{L}$ of complexity $O(1)$ and receiving no input is trivial in the sense that there is at least one output symbol γ such that labeling every node with γ produces a $\mathcal{L}$-legal labeling [2]. Whether or not such γ exists is readily observed from the set of locally consistent labelings of $\mathcal{L}$. If such γ does not exist, the problem's time complexity is $\Omega(\log^* n)$. A theorem by Suomela, published by Chang and Pettie [2] and based on the order-invariance theorem of Naor and Stockmeyer[7] follows.

Theorem 2. *Let $\mathcal{L}$ be any* LCL *problem on a toroidic grid that does not accept input. The time complexity of $\mathcal{L}$ is $O(1)$ if and only if there is an output symbol γ such that every node outputting γ results in a $\mathcal{L}$-legal labeling.*

Proof. Let G be a grid with dimensions $n \times n$. As the identifiers of V_G are bound to a length of $O(\log n)$, the set of identifiers can be considered to be $\{1, 2, \ldots, n^k - 1\}$ for some constant k.

Consider the following way to create such identifiers. Each $u \in V$ has a coordinate (x_u, y_u) such that $x, y \in \{0, 1, \ldots, n-1\}$. Let ϕ_x and ϕ_y be functions mapping integers from $\{0, 1, \ldots, n-1\}$ to integers in $\{0, 1, \ldots, n^k - 1\}$ with the additional requirement that the values of the functions respect the following ordering:

$$\phi_x(0) < \phi_x(1) < \ldots \phi_x(n-1) < \phi_y(0) < \phi_y(1) < \cdots < \phi_y(n-1)$$

. The functions ϕ_x and ϕ_y are used to assign identifiers by setting $\eta(u) = \phi_x(x_u) \cdot n^k + \phi_y(y_u)$ for each node $u \in V_G$.

Due to how the identifiers are assigned, all the identifiers in $B_G(u, t)$ for a node u can now be determined by knowing only $4t + 2$ distinct values: the values of $\phi_x(i)$ when $x_u - t \leq i \leq x_u + t$ and the values of $\phi_y(j)$ when $y_u - t \leq j \leq y_u + t$.

Suppose the complexity of $\mathcal{L}$ is $o(\log^* n)$, and A is a distributed algorithm that solves $\mathcal{L}$ in $t = o(\log^* n)$ communication rounds. Since the correctness of the algorithm is independent of the identifier assignment and A is correct by assumption, it will solve $\mathcal{L}$ correctly when the identifiers are assigned to V_G according to the function η.

Let $u \in V_G$ be a node such that the coordinates (x_u, y_u) are sufficiently far from the "edge" of the grid, where the coordinates wrap around. For this, it is assumed without loss of generality that G is large enough for $n > 2r + 1$ where r is the checking radius of $\mathcal{L}$. Then such a node must exist with $t + r \leq x_u \leq (n-1) - (t + r)$ and $t + r \leq y_u \leq (n-1) - (t + r)$.

Let $S = (s_1, s_2, \ldots, s_{4t+2})$ be the set of values to be used for ϕ_x and ϕ_y, presented as a vector of numbers in $\{0, 1, \ldots, n^k - 1\}$ and ordered in strictly increasing order. Let $\phi_x(\alpha - t - 1 + i) = s_i$ for each $i \in \{1, 2, \ldots, 2t + 1\}$

and $\phi_y(\beta - \tau - 1 + j) = s_{j+2t+1}$ for $j \in \{1, 2, \ldots, 2t+1\}$. Since the A is deterministic, no input is assumed and the edge labels are fixed for the grid, the output depends only on the identifier assignment, and therefore only on the values assigned for S.

Let Γ be the set of output symbols of $\mathcal{L}$. Due to the unique identifier assignment, A can be considered to be a function from $[S]^p$ to Γ. Let $c = |\Gamma|$, $p = 4t + 2$ and $m = 4t + 4r + 2$. Applying the Ramsey theorem, we have $R = R(p, m, c)$. Chang and Pettie [2] present that when p is bound by $t = o(\log^* n)$, $R(p, m, c) < n^k$. Therefore there exists a set $S \subseteq \{0, 2, \ldots, n^k - 1\}$ with $|S| = m = 4t + 4r + 2$ numbers that, when used as identifiers, result in A labeling every node with the same $\gamma \in \Gamma$. As A is correct by assumption, such labeling is $\mathcal{L}$-legal. □

5.2 Computing maximal independent set locally

The problem of finding a *maximal independent set* (MIS) is a simple graph problem that is of fundamental importance. For a graph G, a set $I \subset V_G$ is independent if for all $(u, v) \in E_G$ at least one of $u, v \notin I$. While the property of independence may be interesting as a component of other problems, the problem of finding any independent set is trivial as an empty set is independent.

An independent set I is an MIS if it has the additional property that there exists no $I' \supsetneq I$ that would be independent. Equivalently, an independent set $I \subset V_G$ is an MIS if $I \cup \{w\}$ is not an independent set for any $w \in V_G \setminus I$.

The problem of finding an MIS can be expressed as an LCL problem $\mathcal{L}$. Let the set of output symbols $\Gamma = \{\mathsf{T}, \mathsf{F}\}$ be interpreted so that the set of nodes that output T are considered to be the members of the MIS. The checking radius is defined as $r = 1$, no input is received, and the set of eligible neighborhoods $\mathcal{C}$ is defined to include all output-labeled centered graphs (H, s, λ) of radius 1 so that if $\lambda(s) = \mathsf{T}$, for every $u \in V_H \setminus \{s\} \lambda(u) = \mathsf{F}$. If $\lambda(s) = \mathsf{F}$, at least one $u \in V_H \setminus \{s\}$ must exist with $\lambda(u) = \mathsf{T}$.

To establish that any $\mathcal{L}$-legal labeling is a maximal independent set, consider first the local conditions that would violate the condition of indepen-

dence: two nodes that output T and are connected by an edge. These are not $\mathcal{L}$-legal labelings, as $\mathcal{C}$ contains no legal neighborhood with two adjacent nodes labeled with T. It follows that any $\mathcal{L}$-legal labeling must be an independent set. Now consider a case where the independent set I produced by a $\mathcal{L}$-legal labeling would not be maximal. This means there is a node u labeled F such that $I \cup \{w\}$ is an independent set. This can only be the case if u has no neighbor labeled T, and therefore both u and its neighbors are labeled F: such labeling is not permitted by $\mathcal{L}$. Therefore $\mathcal{L}$ is equivalent to the problem of finding a MIS.

Linial established a lower bound of $\Omega(\log^* n)$ for the time complexity of computing a MIS for arbitrary graphs [6]. For grids and other graphs of a bounded degree, an algorithm that computes a MIS of time complexity $O(\log^* n)$ was introduced by Schneider and Wattenhofer [9]. The following proof is a simpler version, where the grid is first colored with a number of colors bounded by a constant, and the coloring is used to compute the MIS. The coloring algorithm used was introduced by Goldberg et al. [4] and is based on the work of Cole and Vishkin [3].

5.2.1 Distributed tree coloring

An oriented tree is a connected graph G with the following properties. There is a single root $r \in V_G$. For every $u \in V \setminus \{r\}$, u has a parent node denoted as $\pi(u)$. For any distinct $u, v \in V$, $(u, v) \in E_G$ if and only if either $\pi(u) = v$ or $\pi(v) = u$. For the purposes of the algorithm, the nodes are able to recognize which of their edges leads towards their parent. It follows that r initially knows that it is the root of the tree as it is the sole with no parent.

Graph coloring problems are a family of LCL problems whose accepted solutions, *legal colorings*, have each node output a label, customarily referred to as a "color", such that no two neighboring nodes have the same color. Typically, the solution is desired to include as few distinct colors as possible. The algorithm used here produces a legal coloring consisting of only a constant number of colors for a directed tree. For the purposes of describing the algorithm, let φ_u denote the color of the node u. The number of bits required

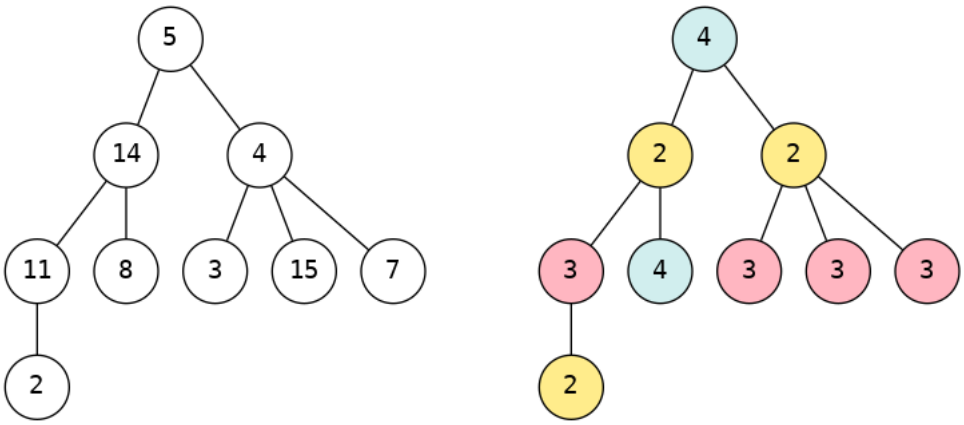

Figure 4: A single color reduction step reducing the coloring of a small tree from nine to three distinct colors while preserving the legality of the coloring.

to represent the color will be denoted as $|\varphi_u|$ and the ith most significant bit of it, for $1 \leq i \leq |\varphi_u|$ will be denoted as $\varphi[i]$.

The algorithm works by beginning with a large number of distinct colours and iteratively performing a constant-time *color reduction* step that re-assigns each node a new color from a smaller set based on the node's prior color and the color of its parent. Initially, each node $u \in V$ sets $\varphi_u = \mathrm{id}(u)$. Each $u \in V$ sends φ_u to each of its children, if any. Each recipient node $v \in V \setminus \{r\}$ will then compare φ_v and $\varphi_{\pi(v)}$ to derive a new color.

The new color for node $v \neq r$ is composed of a bit string α_v and a single bit β_v determined as follows. First, v finds the least i for which $\varphi_v[i] \neq \varphi_{\pi(v)}[i]$, and sets α_v to equal the binary representation of i. The value of β_v is set to equal the bit $\varphi_v[i]$. The new color φ'_v is set as $\alpha_v \oplus \beta_v$ where $\oplus$ denotes the concatenation of two bit strings. The root r has no parent, and chooses the index i arbirarily such that $1 \leq i \leq |\varphi_r|$, but otherwise assigns its new color φ'_r in a similar manner. An example of color reduction is shown in Figure 4.

Theorem 3. *If φ is a legal coloring, φ' is a legal coloring.*

Proof. For any $(u, v) \in E_G$, one of the nodes must be the parent of another, so suppose without loss of generality that $u = \pi(v)$. As φ is a legal coloring

$\varphi_u \neq \varphi_v$. If $\alpha_u = \alpha_v$, by the choice of i, $\varphi'_u[i] \neq \varphi'_v[i]$. Therefore, a parent and child will always differ in either the α or the β component of their color. As β is a single-bit string, it follows that $\varphi'_u \neq \varphi'_v$. □

The initial coloring for each $u \in V_G$ based on $\mathrm{id}(u)$ is clearly a legal coloring, as each node is guaranteed to have an unique identifier. Therefore, iteratively repeating the color reduction step as described will result in a legal coloring no matter how many times it is repeated.

For every $j \in \{1, 2, \ldots\}$ let N_j denote the maximum number of bits required to represent φ_u for any $u \in V$ after j color reduction steps, and $N_0 = \lceil \log n \rceil$ denote the number of bits used before the first color reduction step. For any node u, as α_u is bounded by $|\varphi_u|$ during any color reduction step and β_u is a single bit, $N_{j+1} \leq \lceil \log N_j \rceil + 1 \leq \log N_j + 2$. For instance, $N_1 \leq \log N_0 + 2$ and $N_2 \leq \log(\log N_0 + 2) + 2 \leq \log^2(N_0 + 3)$ when $\log N_0 \geq 2$.

In general, it can be seen that for $j = 1, 2, \ldots$ such that $\log^j N_0 \geq 3$, the inequality $N_j \leq \log^j N_0 + 3$ holds. By the definition of $\log^*$, it follows that when $j = \log^*$, $N_j \leq 5$. As the lengths of the initial identifiers are bounded by $\lceil \log n \rceil$, after $\log^* n$ the number of bits used for the color $|\varphi_u|$ for any $u \in V$ is at most 5. The following two iterations will result in the bit count N_{j+2} being 3. The number of distinct colors is 6 due to α_u having only three possible values 01, 10 and 11 for any node u.

5.2.2 Extension to directed grids

The Cole-Vishkin algorithm for coloring distributed trees can be extended into coloring any graph G where $\deg(u) \leq \Delta$ for all $u \in V_G$ and some constant Δ, such as directed grids where $\Delta = 4$. As an useful note, the method is also applicable to power graphs of grids due to them likewise having a bounded degree for a given power, enabling coloring them as well.

The first step is to perform a forest decomposition on the graph $G = (V, E)$ into $\mathcal{F} = (F_1, F_2, \ldots, F_\Delta)$ where each F_i is a forest, or a set of directed trees. The method described here is by Panconesi and Rizzi [8]. First, each node orients its edges: for $u \in V$ and each $v \in B_G(u, 1) \setminus \{u\}$, u considers the edge (u, v) to be *outbound* if $\mathrm{id}(u) < \mathrm{id}(v)$, *inbound* otherwise. Each node assigns

each outbound edge a distinct forest label from $\{1, 2, \ldots\}$. The subgraph formed by the edges assigned the forest label i will be denoted as F_i. The maximum amount of outbound edges from any node is Δ, also limiting the number of forests in the decomposition to at most Δ.

Each connected component in each $F_i \in \mathcal{F}$ is a directed tree, and hence each such component has a single root. Furthermore, each node can belong in up to Δ different forests in $\mathcal{F}$. Given this decomposition into a forest, the normal Cole-Vishkin algorithm is executed in parallel for each $F_i \in \mathcal{F}$ over the next $\log^* n + O(1)$ communication rounds such that each node maintains a separate record of its color in each forest F_i, denoted as $\varphi_i(u)$. As before, $\varphi_i(u) = \mathrm{id}(u)$ for all $u \in V_G$ and $i \in \{1, 2, \ldots, \Delta\}$.

After $O(\log^* n) + 1$ iterations, $\varphi_i(u) \in \{1, 2, 3, 4, 5, 6\}$ for all u where u belongs to the forest F_i. If u is adjacent to no edge labeled j for $j \in 1, 2, \ldots, \Delta$ it can be considered to be a single-node tree within F_j. As such, it can pick an arbitrary color for $\varphi_j(u)$ from $\{1, 2, 3, 4, 5, 6\}$. With the component colors determined, the ultimate coloring of each $u \in V$ is defined as $\varphi(u) = \langle \varphi_1(u), \varphi_2(u), \ldots, \varphi_\Delta(u) \rangle$.

Theorem 4. *φ is a legal 6^Δ-coloring.*

Proof. For each $i \in \{1, 2, \ldots, \Delta\}$ F_i is a forest consisting of directed trees. Therefore, by the earlier proof, $\varphi_i(u)$ is a legal coloring for all i. For each $(u, v) \in E$ there exists some $i \in \{1, 2, \ldots, \Delta\}$ for which $(u, v) \in E_{F_i}$. Therefore $\varphi_i(u) \neq \varphi_i(v)$. As each color is a three-bit string, the only way for two nodes to share the same color is for them to share all component colors, ensuring that two adjacent nodes will always differ in at least one of their component colors.

Each component color is created by the Cole-Vishkin algorithm as described above, and therefore the number of distinct colors for each component color is 6. For Δ components, this puts the number of distinct colors used at 6^Δ. □

5.2.3 Conversion to a maximal independent set

Consider a graph G and a legal k-coloring $\varphi : V \to \{1, 2, \ldots, k\}$. A maximal independent set can be constructed based on the coloring as follows. Let $K_i = \{u \in V : \varphi_u = i\}$ for $i \in \{1, 2, \ldots, k\}$. Then define $S_1 = K_1$ and S_i for $i \in \{2, 3, \ldots, k\}$ recursively as

$$S_i = S_{i-1} \cup \Big(K_i \setminus B_G(S_{i-1})\Big)$$

.

Theorem 5. *S_k is a maximal independent set.*

Proof. The following inductive argument demonstrates that S_k is an independent set. For every $u \in S_1$, the color $\varphi_u = 1$. As φ is a legal coloring, it follows that there exists no $u, v \in S_1$ such that $(u, v) \in E_G$. Therefore S_1 is an independent set.

For the induction step, suppose S_{j-1} is an independent set. Therefore, $(u, v) \notin E_G$ for any $u, v \in S_{j-1}$. For any $u, v \in K_j$, due to the definition of the set, $\varphi_u = \varphi_v$ and therefore $(u, v) \notin E_G$. Finally, for any $u \in S_j \setminus S_{j-1}$ it holds that $(u, v) \notin E$ for all $v \in S_{j-1}$ due to the construction of S_j excluding the neighborhood of S_{j-1}. Therefore S_j is an independent set.

It remains to be shown that S_k is maximal, or in other words that for every $u \notin S_k$ a $v \in B_G(u, 1)$ exists such that $v \in S_k$. Let $\varphi_u = j$ for some $j \in \{2, 3, \ldots, k\}$. Note that $S_j \subseteq S_k$ and therefore $u \notin S_j$. From the definition of S_1 it follows that $\varphi_u \neq 1$. Then by the definition of S_j at least one $v \in S_{j-1}$ must exist such that $v \in B_G(u, 1)$. □

5.3 Speed-up lemma and the normal form

The speed-up lemma is a result by Brandt et al. [1] establishing a time complexity gap between the complexity classes $\Theta(\log^* n)$ and $\Theta(n)$ in grids. This is accomplished by a method that, given a distributed algorithm A that solves an LCL $\mathcal{L}$ in $T(n) = o(n)$ rounds, produces a new distributed algorithm A' that solves $\mathcal{L}$ in $\Theta(\log^* n)$ communication rounds.

The speed-up lemma is applied by dividing the grid into grid segments of $k \times k$ nodes, and executing A as if each grid segment was a grid in its own right. Let k be the least even integer such that $k \geq 4$ such that $T(k) < k/4-4$. An eligible value exists by the assumption that $T(n) = o(n)$, and the value of k is a constant depending only on T.

Let I be a maximal independent set in $G^{(k/2)}$. This set can be computed in $O(\log^* n)$ communication rounds. The nodes of I will be referred to as anchors. A Voronoi tiling K is constructed for G as follows. For each $v \in I$, let $K(v) = \{u \in V_G : v \text{ is the anchor nearest to } u\}$. Due to I being a maximal independent set in $G^{(k/2)}$ and k being constant, each $u \in V_G$ is at most $k/2$ edges away from the nearest anchor and can therefore compute which tile they belong to in constant time. Breaking ties to determine which tile a node belongs to when two or more anchors are equally distant from a given node can be done arbitrarily.

Using the Voronoi tiles, each node can be assigned a new locally unique identifier using its relative location to the anchor. For $v \in I$ and $u \in K(v)$, let $c(u) = (x_u - x_v, y_u - y_v)$. Then $c(u)$ will be used as the identifier for each node u. As any distinct $i, i' \in I$ have $\text{dist}_G(i, i') > (k/2)$ and the identifier assignment is determined by relative position to anchor, any $u, v \in V_G$ with $c(u) = c(v)$ will also have $\text{dist}_G(u, v) > (k/2)$.

Now, A is applied to G but using the relative coordinates as identifiers, as if A was locally solving a $k \times k$ instance of $\mathcal{L}$. By assumption A completes in $T(k) < k/4$ communication rounds, and therefore will never see repeating identifiers. The output of A executed in this way is correct by our assumption that A is correct: if A produces an incorrect solution for G, it would produce an incorrect output on an identically labeled normal grid of dimensions $k \times k$, contradicting the assumption.

As k depends only on the time complexity of $\mathcal{L}$, it is a constant for a given problem. Therefore executing A for an instance of size $k \times k$ has a time complexity of $O(1)$. In addition to this, the only computation required is to find the maximal independent set for $G^{(k/2)}$, resulting in the sped-up time complexity of $\Theta(\log^* n)$.

5.4 Undecidability of time complexity classification

While LCL problems of time complexity of $O(1)$ on grids are easy to recognize, determining whether a given LCL belongs to the time complexity class $\Theta(\log^* n)$ or $\Theta(n)$ is undecidable. The following proof is by Brandt et al [1].

Theorem 6. *The problem of determining whether a given* LCL $\mathcal{L}$ *can be solved in* $O(\log^* n)$ *communication rounds on a toroidic grid is undecidable.*

Proof. For each Turing machine M, let $\mathcal{L}_M$ be an LCL problem whose outputs and acceptable labelings are the disjoint union of two other LCL problems: $\mathcal{P}_1$ and $\mathcal{P}_2$. Any valid solution to $\mathcal{L}_M$ is therefore a valid solution to either $\mathcal{P}_1$ or $\mathcal{P}_2$. Let $\mathcal{P}_1$ be the 3-coloring problem, regardless of the choice of M. A legal 3-coloring exists for all $n \times n$ grids except for the degenerate case of 1×1, and therefore $\mathcal{L}_M$ can always be solved for grids of dimensions of at least 2×2.

However, the time complexity of 3-colouring is known to be $\Omega(n)$ [1], which means that any faster solution must be a legal solution to $\mathcal{P}_2$. The problem $\mathcal{P}_2$ will be the problem of dividing the grid into *patches* of a size bounded by a constant s depending only on the choice of M. Each patch will be labeled in a way that encodes the execution table of M when started on an empty tape. The problem will be specified in a manner such that it can be solved in $O(\log^* n)$ communication rounds if and only if M halts in finite time.

5.4.1 Defining $\mathcal{L}_M$

In a legal labeling for $\mathcal{P}_2$ each node is belongs to one of the three classes *border*, *quadrant* or *anchor*.

Each node is associated with a type label, with the eligible type labels being $\{\mathsf{N}, \mathsf{E}, \mathsf{S}, \mathsf{W}\}$ for borders, $\{\mathsf{NE}, \mathsf{SE}, \mathsf{SW}, \mathsf{NW}\}$ for quadrants and A for anchors. The type label of a node u is denoted as $Q(u)$.

A *diagonal neighbor* of a node u with $Q(u) \neq \mathsf{A}$ is the node reached from u by traveling in the direction indicated by $Q(u)$, and is denoted as $\mathrm{diag}(u)$. Expanding the normal notation for neighborhood relationships in grids,

$\mathsf{NE}(u) = \mathsf{N}(\mathsf{E}(u))$,

$\mathsf{SE}(u) = \mathsf{S}(\mathsf{E}(u))$,

$\mathsf{NW}(u) = \mathsf{N}(\mathsf{W}(u))$,

$\mathsf{SW}(u) = \mathsf{S}(\mathsf{W}(u))$.

Additionally, if $Q(u) = \mathsf{A}$, $\text{diag}(u) = u$.

The desired labeling of quadrants in $\mathcal{P}_2$ creates large continuous quadrant labeled areas where following the grid in the direction indicated by the quadrant labels eventually leads to a correctly labeled border or anchor. Let the following constraints be defined:

when $Q(v) = \mathsf{NE}$ then $Q(\text{diag}(v)) \in \{\mathsf{NE}, \mathsf{N}, \mathsf{E}, \mathsf{A}\}$,

when $Q(v) = \mathsf{SE}$ then $Q(\text{diag}(v)) \in \{\mathsf{SE}, \mathsf{S}, \mathsf{E}, \mathsf{A}\}$,

when $Q(v) = \mathsf{SW}$ then $Q(\text{diag}(v)) \in \{\mathsf{SW}, \mathsf{S}, \mathsf{W}, \mathsf{A}\}$, and

when $Q(v) = \mathsf{NW}$ then $Q(\text{diag}(v)) \in \{\mathsf{NW}, \mathsf{N}, \mathsf{W}, \mathsf{A}\}$.

For a border node u, $\mathcal{P}_2$ shall specify that either $Q(\text{diag}(u)) = Q(u)$ or $Q(\text{diag}(u)) = \mathsf{A}$. In addition, the border must be placed between the correct quadrants as follows:

when $Q(u) = \mathsf{N}$ then $Q(\mathsf{W}(u)) = \mathsf{NE}$ and $Q(\mathsf{E}(u)) = \mathsf{NW}$,

when $Q(u) = \mathsf{E}$ then $Q(\mathsf{N}(u)) = \mathsf{SE}$ and $Q(\mathsf{S}(u)) = \mathsf{NE}$,

when $Q(u) = \mathsf{S}$ then $Q(\mathsf{W}(u)) = \mathsf{SE}$ and $Q(\mathsf{E}(u)) = \mathsf{SW}$, and

when $Q(u) = \mathsf{W}$ then $Q(\mathsf{N}(u)) = \mathsf{SW}$ and $Q(\mathsf{S}(u)) = \mathsf{NW}$.

To ensure each quadrant is aligned correctly around the anchor, $\mathcal{P}_2$ also specifies that for any u with $Q(u) = \mathsf{A}$, all of the following hold: $Q(\mathsf{N}(u)) = \mathsf{S}$, $Q(\mathsf{NE}(u)) = \mathsf{SW}$, $Q(\mathsf{E}(u)) = \mathsf{W}$, $Q(\mathsf{SE}(u)) = \mathsf{NW}$, $Q(\mathsf{S}(u)) = \mathsf{N}$, $Q(\mathsf{SW}(u)) = \mathsf{NE}$, $Q(\mathsf{W}(u)) = \mathsf{E}$, and $Q(\mathsf{NW}(u)) = \mathsf{SE}$.

Nothing in the constraints of $\mathcal{P}_2$ forces the existence of an anchor, so it is possible for a legal solution of $\mathcal{P}_2$ to fill the entire grid with the same

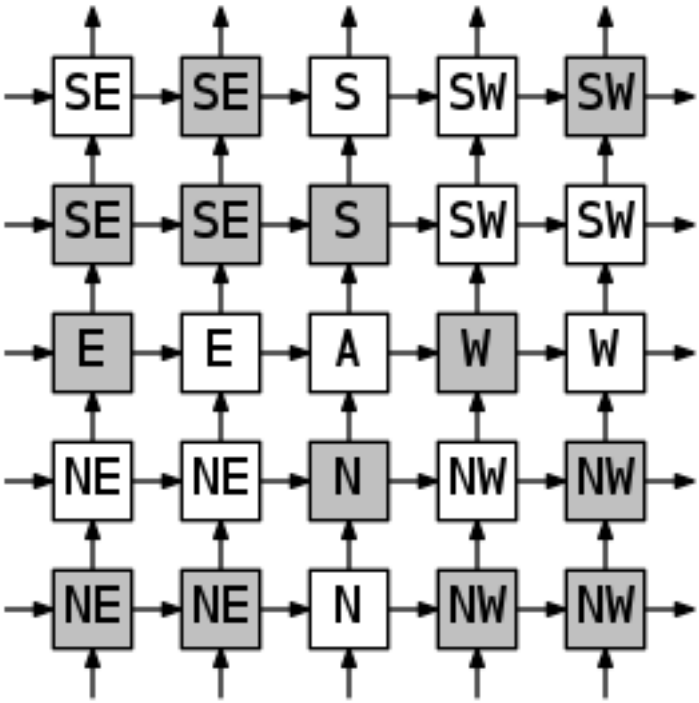

Figure 5: An example of the surroundings around an anchor of $\mathcal{P}_2$. Each node's label $Q(u)$ points towards the nearest anchor. The shading indicates the diagonal 2-coloring. For a Turing machine M, the encoding of its execution table $E(M)$ will be output by the anchor and nodes of type S, W or SW.

quadrant label. To ensure that $\mathcal{P}_2$ is not trivial and that any solution of time complexity $o(n)$ has to place anchors to divide the grid into patches, $\mathcal{P}_2$ additionally requires that that each node outputs a color $c : V \to \{0, 1\}$ such that either $c(u) \neq c(\mathrm{diag}(u))$ or $Q(u) \neq Q(\mathrm{diag}(u))$ for all $u \in V$. As the complexity of 2-coloring the path formed by the diagonal is inherently $\Omega(n)$ the dimensions of any patch must be bounded by a constant in order for $\mathcal{P}_2$ to be computable in $o(n)$ communication rounds.

As a final requirement, any legal solution of $\mathcal{P}_2$ must contain, around each anchor, the entire execution of M starting with an empty tape encoded in the quadrant labeled SW and the borders and anchors around it. This encoding is denoted as $E(M)$. The earlier labeling conditions guarantee that such a quadrant must exist in any solution of complexity $o(n)$.

The constraints for creating $E(M)$ as a labeling in G are as follow. Let each anchor u be associated with coordinates $u = (0, 0)$, and each other node its coordinates relative to the closest anchor. For a node in coordinates (n, m) with non-negative n and m, $E(M)$ must be labeled with the symbol that is in the nth position of the tape of M after the mth step of execution. In addition, the label of such node (n, m) will also include the machine's state when head of M is at position n after executing the mth step of computation, and a null symbol otherwise.

Supposing M halts after s steps and accesses r cells of tape during its execution, he dimensions of the subgrid containing $E(M)$ will be $r \times (s + 1)$. The local constraints of $E(M)$ include that M begins with an empty tape, with the head at the leftmost position on the tape: the anchor node at $(0, 0)$. That $E(M)$ respects the transition rules of M can also be checked locally: observing individual 2×2 subgrids is enough to determine that only the cell containing the head can be written, the head does not disappear, move more than a single step or duplicate, and that all writing, state transitions and movements of the head follow the rules of M. It can also easily be checked locally that no node u with $Q(u) \notin \{\mathsf{S}, \mathsf{W}, \mathsf{SW}\}$ is labeled with the state symbols or tape symbols of M, and that the state symbol for the final row of $E(M)$ represents one of the halting states of M.

In summary, each output label for $\mathcal{P}_2$ is a tuple containing the type label,

a color bit for 2-coloring of the diagonals, one of the states of M or a null symbol, and one of the tape symbols of M or a null symbol. As each of these sets is finite, the output label set of $\mathcal{P}_2$ is likewise finite.

The constraints for a legal labeling for $\mathcal{P}_2$ are that each node has one of the type labels, each diagonal is 2-colored, and each 2×2 subgrid is locally consistent with the specification of $E(M)$. The set of legal neighborhoods is also finite, with $\mathcal{P}_2$ having a checking radius $r = 2$.

A correct solution to $\mathcal{L}_M$ is a correct solution to either $\mathcal{P}_1$ or $\mathcal{P}_2$. As they have been defined to use a disjoint set of output labels, it is trivial to locally check which solution is attempted and that the grid does not contain solution attempts for both problems. $\mathcal{P}_1$ is defined as 3-coloring, so given a solution candidate identified as one for $\mathcal{P}_1$, it suffices to check the outputs in $B_G(u, 1)$ for each $u \in V_G$ and see if any node repeats the output of u. As the encoding of $\mathcal{P}_2$ is likewise locally checkable it follows that $\mathcal{L}_M$ is an LCL.

5.4.2 Solving L_M

Brandt et al. [1] present a way to solve $\mathcal{L}_M$ in $O(\log^* n)$ communication rounds assuming that M halts in finite time when executed with an empty tape. Let s be the number of steps M takes before halting. By assumption, s is a constant depending only on M.

Let the grid G be of dimensions $n \times n$ with $n \geq 4(s + 1)$. Compute a maximal independent set I in $G^{(4(s+1))}$. The nodes of I will be used as anchors, so $Q(u) = \mathsf{A}$ is set for each $u \in I$.

Each $u \in V_G$ finds their closest anchor v and computes their relative coordinates. This results in a Voronoi tiling T of I. As anchors are at a distance of most $4(s+1)$ edges and s is problem-defined constant, the creation of the Voronoi tiling and resolving the local coordinates takes only $O(1)$ time. Each node can then assign themselves a type label as follows. With the anchor v at $(0, 0)$, for each node $u \in T(v)$ at coordinates (x_u, y_u)

$$Q(u) = \begin{cases} \mathsf{N} & \text{when } x_u = 0 \text{ and } y_u < 0, \\ \mathsf{NE} & \text{when } x_u < 0 \text{ and } y_u < 0, \\ \mathsf{E} & \text{when } x_u < 0 \text{ and } y_u = 0, \\ \mathsf{SE} & \text{when } x_u < 0 \text{ and } y_u > 0, \\ \mathsf{S} & \text{when } x_u = 0 \text{ and } y_u > 0, \\ \mathsf{SW} & \text{when } x_u > 0 \text{ and } y_u > 0, \\ \mathsf{W} & \text{when } x_u > 0 \text{ and } y_u = 0, \\ \mathsf{NW} & \text{when } x_u > 0 \text{ and } y_u < 0. \end{cases}$$

Defining the 2-coloring at the same time is simple, as by $4(s+1)$ communication rounds all nodes in the same quadrant or border of $T(u)$ for some $u \in I$ have communicated with each other.

Starting with anchors $u \in I$, the execution of M is simulated in G. As M terminates in s steps by assumption, and anchors have a distance of at least $4(s+1)$, the entire $E(M)$ will fit in one Voronoi tile. Since s is constant for a given choice of M, the creation of $E(M)$ has a time complexity of $O(1)$. The only operation with a non-constant time bound is the calculation of the maximal independent set which takes $O(\log^* n)$ communication rounds. Therefore the time complexity of solving $\mathcal{P}_2$ and by extension $\mathcal{L}_M$ when M halts is also $O(\log^* n)$.

Assuming M does not halt on an empty tape, it can still be solved in $\Omega(n)$ communication rounds by solving $\mathcal{P}_1$. To establish that no faster algorithm for $\mathcal{L}_M$ can exist when M does not halt, two possible outcomes of $\mathcal{P}_2$ for a non-halting M are considered.

The local constraints of $\mathcal{P}_2$ cannot specify that an anchor has to appear in the solution, so a legal solution can exist that has no anchors. Without an anchor there cannot be borders, so a solution is to let each $u \in V_G$ have the same type label from $\{\mathsf{NE}, \mathsf{SE}, \mathsf{SW}, \mathsf{NW}\}$. This is a legal way to solve $\mathcal{P}_2$, but the length of each diagonal will be $\Omega(n)$ and consequently take $\Omega(n)$ rounds to 2-color.

Assume that the solution to $\mathcal{P}_2$ contains at least one anchor. Let $u = (0, 0)$

be an arbitrary anchor. Within the same patch, each node $(0, j)$ for every positive integer j must be assigned the type label S and the symbol label at the first position of the tape of M after the jth step of the execution of M. Similarly, each node $(i, 0)$ for a positive i must be assigned the type label W and empty tape symbols, as M starts on empty tape. Since by assumption M does not halt, one of the following must happen: either $E(M)$ will contain a faulty transition which can be detected locally, or $E(M)$ will continue into another patch or around the grid, in which there will be a node v with $Q(v) \notin \{\mathsf{S}, \mathsf{W}, \mathsf{SW}\}$ that contains a part of the labeling of $E(M)$. As neither outcome is legal for $\mathcal{P}_2$, no legal solution for $\mathcal{P}_2$ can contain an anchor if M does not halt in finite time.

As P_1 requires $\Omega(n)$ communication rounds to solve, and a correct solution to P_2 requires $\Omega(n)$ communication rounds if M does not halt on an empty tape, solving $\mathcal{L}_M$ is likewise a global problem if M does not halt. If M does halt, P_2 can be solved using the method outlined above, thereby solving L_M in $O(\log^* n)$ communication rounds. However, because M can be an arbitrary Turing machine, an accurate time complexity classification for $\mathcal{L}_M$ would solve the halting problem which is known to be undecidable. Therefore the problem of determining whether a given LCL problem is global or local is likewise undecidable. □

6 Algorithm synthesis

While determining the time complexity for an LCL in a toroidic grid is impossible in a general case, a method exists for creating an asymptotically optimal algorithm for an LCL problem whose complexity is already known. [1]

Constructing asymptotically optimal algorithms is simple for problems of complexity $O(1)$ and $\Omega(n)$. In the former case, there exists a constant output that an algorithm can print to satisfy the constraints of the problem [2], and therefore an asymptotically optimal algorithm can simply print such output and halt within a single communication round. For global problems, an asymptotically optimal algorithm is for each node to relay all information available to them until each node knows the structure of the whole graph,

and can then solve the problem locally. Creating optimal algorithms for the remaining complexity class, problems of complexity $\Theta(\log^* n)$, is a more complex process originally introduced by Brandt et al [1], referred to as algorithm synthesis.

Algorithm synthesis for a local LCL $\mathcal{L}$ works by reducing $\mathcal{L}$ to the computation of maximal independent set in some power graph of the underlying grid in a manner similar to the speed-up lemma discussed in Section 5.3. Let MIS_k be an algorithm that, when executed on grid G, labels V_G with a maximal independent set of G^k in $O(\log^* n)$ communication rounds. One such algorithm is presented in Section 5.2. The nodes belonging to this maximal independent set will be referred to as *anchors*. A synthetized algorithm A is of the form $A' \circ \mathsf{MIS}_k$ where k is a problem-dependent constant, and A' is an algorithm that can solve $\mathcal{L}$ in $O(k)$ rounds given the placement of anchors as an input. Therefore, the problem of designing an asymptotically optimal algorithm for an $\mathsf{LCL}$of time complexity $O(\log^* n)$ can be broken into three sub-problems: finding a sufficiently large value for k, enumerating all possible arrangements of anchors in a $O(k)$-radius neighborhood of a node, and constructing A'.

6.1 Forming the tiles

Consider a sufficiently large grid G and $I_k \subset V_G$ which is a maximal independent set of G^k for some $k \geq 1$. A *k-tile* is a pair (T, I_T), where T is an induced subgraph of G and $I_T = I_k \cap V_T$. A k-tile can therefore be seen as a possible local arrangement of anchors for the purposes of algorithm synthesis. In order to construct the mapping A', algorithm synthesis requires the construction of all sufficiently large tiles for a particular k in order to work.

Determining whether a given grid segment labeled with a maximal independent set in its kth power is a possible k-tile is not a trivial task. When determining if a given pair (T, I_T) is a k-tile, cases containing distinct $u, v \in I_T$ with $\mathrm{dist}_T(u, v) \leq k$ can be discarded right away, as then I_T is not independent in T^k.

For the remaining cases, consider the set

$$U = \{u \in V_T : \forall v \in I_T \ \mathrm{dist}_T(u, v) > k\}$$

. The set U represents nodes within V_T that are *unsatisfied* in the sense that they are not close enough to an anchor to guarantee that I_T does not contradict the maximality property of I_k. If $U = \emptyset$, (T, I_T) is a legal k-tile.

As each legal k-tile represents a possible way to arrange anchors in G^k, the property of being a legal k-tile is inherited to subsets of legal k-tiles. Conversely, to ascertain whether (T, I_T) is a legal tile if $U \neq \emptyset$, a larger k-tile containing (T, I_T) can be constructed. If the larger tile is demonstrably legal, (T, I_T) is legal as well.

One approach to enumerating all rectangular k-tiles of dimensions $w \times h$ is as follows. First, let C be the set of all tiles of dimensions $1 \times h$. From the definition of legal k-tile, it follows that each k-tile of dimensions $w \times h$ can be represented as a vector of w k-tiles of dimensions $1 \times h$. Therefore, the enumeration can be represented as a tree where elements of C are iteratively added to increase the width of the tile. Branches where the addition of a particular column would not create an independent set are pruned.

The leaves of this tree are potential k-tiles of dimensions $w \times h$. To establish the legality of each such tile (T, I_T), let the set U represent the unsatisfied nodes within (T, I_T). If $U = \emptyset$, (T, I_T) is a tile. Otherwise, it must be determined whether I_T can be extended outside T to create a larger tile that satisfies each $u \in U$, with each added node still being more than k edges away from every $v \in I_T$ to preserve the property of being an independent set in G^k.

If $U \neq \emptyset$, let each $u \in V_T$ be represented by its co-ordinate pair (x_u, y_u), with $(0, 0)$ representing the node in the north-western corner of the tile. For two nodes u and v, let $d_1(u, v) = (|x_u - x_v|, |y_u - y_v|)$. For each $u \in U$, let the set of nodes outside the candidate tile but eligible to satisfy u be defined as

$$S_u = \{v \notin V_T : d_1(u, v) \leq k\} \setminus \bigcup_{w \in I_T} \{v \notin V_T : d_1(v, w) \leq k\}$$

.

If $S_u = \emptyset$ for any $u \in U$, then u is a node within (T, I_T) that is at a distance of more than k from every node within I_T and cannot be satisfied by an external node, in which case (T, I_T) is not a legal tile. Otherwise, one attempts to find a set $I_S \subseteq \bigcup_{u \in U} S_u$ such that $I_S \cap S_u \neq \emptyset$ for all $u \in U$ and $d_1(v, w) > k$ for every distinct $v, w \in I_S$. If such set can be found, (T, I_T) is a legal tile. The task of finding I_S can be performed efficiently for instance by presenting the problem in conjunctive normal form and feeding it to a SAT solver.

6.2 Reduction step

Let I_k denote an arbitrary maximal independent set for G^k. As the output of A' for a given $u \in V_G$ depends only on the locations of anchors within $B_G(u, k)$, and the number of different arrangements of anchors within a finite-sized neighborhood is also finite, A' can be seen as a simple finite function from all possible relative arrangements of anchors around an arbitrary node to the output set of $\mathcal{L}$ which shall be denoted as Γ.

The domain of A' is the set of all k-tiles of dimensions $w \times h$ for suitably large $w, h = O(k)$, with the codomain being Γ. The output label of each node must be assigned such that adjacent nodes always get labels respecting the constraints of $\mathcal{L}$. This is accomplished by turning the tiles into a grid-like structure, a *neighborhood graph*, and executing A on this graph.

The neighborhood graph $M = (V_M, E_M, \ell_M)$ is constructed as follows. Let V_M be the set of all k-tiles of dimensions $w \times h$ for sufficiently large w, h. For edges, *horizontal* and *vertical* k-tiles are considered. A horizontal k-tile is a k-tile of dimensions $w + 1 \times h$ and a vertical k-tile is of dimensions $w \times h + 1$. For k-tiles $u, v \in V_M$ the node v is considered to be the eastern neighbor of u if there exists a horizontal k-tile whose w westernmost columns equal u and whose w easternmost columns equal v. Similarly, v is considered to be the northern neighbor of u if there exists a vertical k-tile whose h southernmost rows equal u and whose h northernmost rows equal v. The edge labeling ℓ_M is set such that an edge e orienting nodes towards their northern neighbors

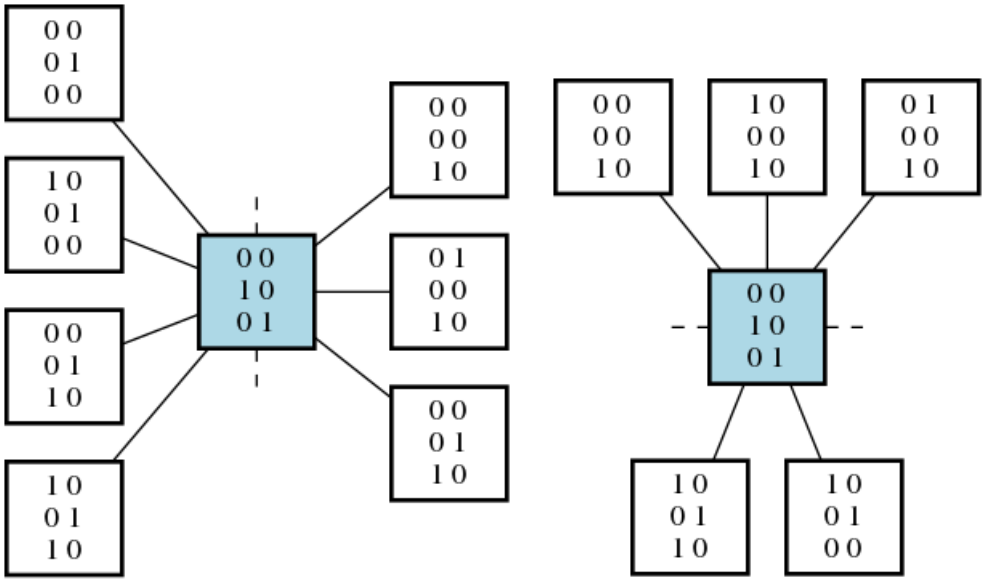

Figure 6: The neighbors of a 1-tile of dimensions 2×3, with horizontal neighbors presented on the left and vertical neighbors on the right. Note that some of the tiles are neighbors of the center tile in more than one direction.

has $\ell_M(e) = \mathcal{N}$ and $\ell_H(e^R) = \mathcal{S}$. Similarly, an edge e between a node and its eastern neighbor has $\ell_M(e) = \mathcal{E}$ and $\ell_M(e^R) = \mathcal{W}$.

The construction of the neighborhood graph brings forth a special property that requires altering the normal conventions of graphs: for k-tiles $u, v \in V_M$, it is possible that according to the definitions above, that u and v are each others' neighbors in more than one direction when $k = 1$. Each of these relationships is important individually, so let the set of edge labels for such case be defined as $\mathcal{P}(\{\mathcal{N}, \mathcal{E}, \mathcal{S}, \mathcal{W}\})$. An edge labeled with a set containing more than one direction is considered to be oriented according to each of those directions for the purposes of satisfying LCL constraints.

As an example of k-tile generation and neighborhood graph construction, consider 1-tiles of dimensions 2×3. Using 1 to denote anchors and 0 to denote non-anchors, the legal tiles are the following:

$$\begin{array}{|c|}\hline 00\\00\\10\\\hline\end{array}\ \begin{array}{|c|}\hline 00\\00\\01\\\hline\end{array}\ \begin{array}{|c|}\hline 00\\10\\00\\\hline\end{array}\ \begin{array}{|c|}\hline 00\\01\\00\\\hline\end{array}\ \begin{array}{|c|}\hline 10\\00\\00\\\hline\end{array}\ \begin{array}{|c|}\hline 01\\00\\00\\\hline\end{array}\ \begin{array}{|c|}\hline 00\\01\\10\\\hline\end{array}\ \begin{array}{|c|}\hline 10\\00\\10\\\hline\end{array}$$

$$\begin{array}{|c|}\hline 01\\00\\10\\\hline\end{array}\ \begin{array}{|c|}\hline 00\\10\\01\\\hline\end{array}\ \begin{array}{|c|}\hline 10\\00\\01\\\hline\end{array}\ \begin{array}{|c|}\hline 01\\00\\01\\\hline\end{array}\ \begin{array}{|c|}\hline 01\\10\\00\\\hline\end{array}\ \begin{array}{|c|}\hline 10\\01\\00\\\hline\end{array}\ \begin{array}{|c|}\hline 01\\10\\01\\\hline\end{array}\ \begin{array}{|c|}\hline 10\\01\\10\\\hline\end{array}$$

Each legal 1-tile of dimensions 3×3 then defines a horizontal edge in E_M. For example,

$$\begin{array}{|c|}\hline 100\\010\\101\\\hline\end{array}\ \text{is a legal 1-tile and therefore}\ \left(\begin{array}{|c|}\hline 10\\01\\10\\\hline\end{array}\ ,\ \begin{array}{|c|}\hline 00\\10\\01\\\hline\end{array}\right) \in E_M.$$

Similarly, each legal 1-tile of dimensions 2×4 defines a vertical edge in E_M. For example,

$$\begin{array}{|c|}\hline 10\\01\\10\\00\\\hline\end{array}\ \text{is a legal 1-tile and therefore}\ \left(\begin{array}{|c|}\hline 01\\10\\00\\\hline\end{array}\ ,\ \begin{array}{|c|}\hline 10\\01\\10\\\hline\end{array}\right) \in E_M.$$

One should note that there are cases where mating 1-tiles together by their overlapping rows or columns does not define an edge. For instance,

$$\begin{array}{|c|}\hline 101\\000\\101\\\hline\end{array}\ \text{is not a valid 1-tile and therefore}\ \left(\begin{array}{|c|}\hline 10\\00\\10\\\hline\end{array}\ ,\ \begin{array}{|c|}\hline 01\\00\\01\\\hline\end{array}\right) \notin E_M.$$

Now M is a graph that describes every possible k-tile of dimensions $w \times h$ and every way they can appear adjacent to each other. M resembles a grid

in the sense that nodes are adjacent to each other in an oriented fashion: however, each node can have multiple neighbors in any given direction, and even have multiple differently oriented edges towards the same neighbor. Solving $\mathcal{L}$ on M produces a desired mapping A' from k-tiles to Γ. Since M is not a grid, however, the detail of encoding the constraints of $\mathcal{L}$ must be addressed.

The model of encoding used within this thesis is encoding the constraints of $\mathcal{L}$ to concern only individual edges, and is implemented as follows. Let $\mathcal{C}$ be the set of locally consistent labelings of $\mathcal{L}$. Assume w.l.o.g. that the checking radius of $\mathcal{L}$ is 1. Then every $c \in \mathcal{C}$ consists of five labeled nodes: a center node and one node in each direction of north, east, south and west. Let Γ be the set of eligible output symbols of $\mathcal{L}$. It can be seen that $|\mathcal{C}| \leq |\Gamma|^5$. For convenience, let $\mathcal{C}_U$ denote the set of all labeled radius-1 neighborhoods labeled with elements of Γ, including ones that are not locally consistent labelings of $\mathcal{L}$.

For any $c \in \mathcal{C}_U$, let $\mathsf{C}(c)$ denote the label in Γ that is the label of the center node of c. Similarly, let $\mathsf{N}(c)$, $\mathsf{E}(c)$, $\mathsf{S}(c)$ and $\mathsf{W}(c)$ denote the element in Γ that is the label of the northern, eastern, southern and western node of c, respectively. For any $c, c' \in \mathcal{C}$, let (c, c') be a *consistent horizontal edge* if $\mathsf{C}(c) = \mathsf{W}(c')$ and $\mathsf{E}(c) = \mathsf{C}(c')$, and a *consistent vertical edge* if $\mathsf{C}(c) = \mathsf{S}(c')$ and $\mathsf{N}(c) = \mathsf{C}(c')$. A pair including any element from $\mathcal{C}_U \setminus \mathcal{C}$ is never a consistent edge.

Next, a function $g : V_M \to \mathcal{C}$ must be found such that the following is true. For every $(u, v) \in E_M$ with $\ell_M(u, v) = \mathcal{N}$, (u, v) must be a consistent vertical edge, and for every $(u, v) \in E_H$ with $\ell_H(u, v) = \mathcal{E}$, (u, v) must be a consistent horizontal edge. Considering the southern and western directions is not necessary as the conditions for horizontal and vertical consistency guarantee consistency in the opposite direction as well. If such g can be found, $\mathcal{L}$ is solved on M by setting the output of each $u \in V_M = \mathsf{C}(g(u))$.

The task of finding the mapping g can be performed by converting the constraints of the problem to a boolean satisfiability problem in conjunctive normal form, after which a SAT solver can produce the mapping. Let $f_V : V_M \to \{0, 1, 2, \ldots, |V_M| - 1\}$ and $f_{\mathcal{C}} : \mathcal{C}_U \to \{1, 2, \ldots, |\mathcal{C}_U|\}$ be arbitrary

bijections. Note that $|\mathcal{C}_U| = |\Gamma|^5$. Let $n = \log |\Gamma|^5$, or the number of bits required to represent each distinct output neighborhood. Now, for each node $u \in V_H$, n variables are assigned in the following manner. Let $f_V(u) = i$. Then x_{ni+1}, x_{ni+2}, ..., x_{ni+n} are variables that together encode the possible output labels received by u. For an output label $c \in \mathcal{C}_U$, let c_j represent the jth most significant bit of $f_{\mathcal{C}}(c)$. To represent u receiving the output label c, the variables are set as

$$x_{ni+j} = \begin{cases} True, & \text{if } c_j = 1 \\ False, & otherwise \end{cases} \tag{1}$$

.

With the encoding for a node receiving the particular output label fixed, the task of finding a legal assignment of output labels must still be defined. An easy way to perform this in conjunctive normal form accepted by SAT solvers is to enumerate all configurations that can *not* be seen in a legal assignment. For every $(u, v) \in E_M$ with $\ell_M(u, v) = \mathcal{E}$, and every $c, c' \in \mathcal{C}_U$ such that (c, c') is not a consistent horizontal edge, let the following clause be added. With $i = f_V(u)$ and $i' = f_V(v)$, let there be a CNFSAT clause containing the literals, x_{ni+j} if $c_j = 0$ and $\neg x_{ni+j}$ otherwise, and $x_{ni'+j}$ if $c'_j = 0$ and $\neg x_{ni'+j}$ otherwise, for each $j \in \{1, 2, \ldots, n\}$. Informally, each such clause constrains a pair of nodes to avoid a particular non-consistent labeling between each other. With all non-consistent labelings ruled out, M is labeled by g such that all edges in $B_M(u, r)$ are consistent for all $u \in V_M$ and $r = 1$.

With the function g and the value of k thus found, $\mathcal{L}$ can finally be solved in $O(\log^* n)$ communication rounds for the grid G by computing a MIS I in $O(\log^* n)$ communication rounds, then over the course of $O(k)$ communication rounds each node determines what k-tile $T \in V_M$ matches the arrangement of anchors within their neighborhood. The node then sets its output as $\mathsf{C}(g(T))$.

The length of each clause is $2n$. As each clause corresponds to a particular edge in E_M labeled with symbols $c, c' \in \mathcal{C}_U$ and $|\mathcal{C}_U| = |\Gamma|^5$, the total number of clauses needed in this way is bounded by $O(|E_M| \cdot |\Gamma|^5)$.

7 Searching for local problems

While determining whether an arbitrary LCL problem is global or not in a toroidic grid is undecidable, algorithm synthesis can be applied as a tool for recognizing local problems efficiently. By automatizing this process, suitably encoded LCL problems can be fed to a solver in bulk with a suitably large fixed value of k, in order to identify large numbers of problems of sub-global complexity. An implementation of the methods presented in this thesis will be made available in the author's GitHub [10].

In order to narrow the search space and computational complexity of searching for local problems, attention is restricted to a family of LCL problems that will be referred to as *quaternary edge* problems. This family contains only the LCL problems that satisfy the following criteria. The checking radius of each problem is 1, the problems accept no input, and the set of output symbols is $\Gamma = \{0, 1, 2, \ldots, 15\}$ represented internally with four bits. Furthermore, to narrow the search space and reduce the complexity of implementation, each problem must be specifiable in terms of *legal edges*.

In terms of more general LCL problems, the set of locally consistent labelings of a quaternary edge problem is the set of radius-1 neighborhoods where each pair of nodes connected by an edge adhere to a legal labeling. Each legal edge can be represented as a tuple $(\gamma, \gamma', \underset{\cdot}{)}$ where $\gamma, \gamma' \in \Gamma$ and $d \in \{\mathcal{N}, \mathcal{E}\}$. Note that specifying southbound and westbound edges separately is not needed, as they can be considered to be mirrored from the northbound and eastbound ones. The set of legal edges for any quaternary edge problem $\mathcal{L}$ will be denoted as $\mathcal{C}^E(\mathcal{L})$. As $|\Gamma| = 4$ and $|d| = 2$, the number of distinct labeled directional edges is $4^2 \cdot 2 = 32$.

As the input symbols, output symbols and checking radius are fixed, the only difference between any two quaternary LCL probems is their set of locally consistent labelings. Since there are 32 distinct legal edges and any subset of them is a distinct LCL problem, there are 2^{32} distinct LCL problems to consider. This is slightly over four billion, a number unfeasible for practical computation. However, methods that can be used to prune the set of problems will be introduced.

7.1 Superset pruning

The first major pruning strategy is to observe that the time complexity of a graph problem can only decrease as legal edges are added.

Theorem 7. *Let $\mathcal{A}$ and $\mathcal{B}$ be quaternary* LCL *problems such that $\mathcal{C}^E(\mathcal{A}) \subseteq \mathcal{C}^E(\mathcal{B})$. Then if $\mathcal{A}$ is local, $\mathcal{B}$ is also local.*

Proof. As $\mathcal{A}$ is local by assumption, an algorithm that computes it in $O(\log^* n)$ communication rounds must exist. Let this algorithm be denoted as A. Let G be an arbitrary grid and l be the labeling produced by A when executed on G. Let $(u, v) \in E_G$ and $d \in \{\mathcal{N}, \mathcal{E}\}$ be any edge and its direction in G. Then $(l(u), l(v), d) \in \mathcal{C}^E(\mathcal{A})$ as A solves $\mathcal{A}$ correctly by assumption. Since $\mathcal{C}^E(\mathcal{A}) \subseteq \mathcal{C}^E(\mathcal{B})$, it follows that $(l(u), l(v), d) \in \mathcal{C}^E(\mathcal{B})$. Therefore A solves $\mathcal{B}$ in $O(\log^* n)$ communication rounds. □

As a consequence, one can attempt algorithm synthesis on hard problems first in an effort to rule out larger sections of the entire family of quaternary LCL problems.

Furthermore, a similar argument can be applied to show that if $\mathcal{A}$ is trivial, so is $\mathcal{B}$. Therefore, any quaternary LCL problem $\mathcal{L}$ with $(\gamma, \gamma, \mathcal{N}), (\gamma, \gamma, \mathcal{E}) \in \mathcal{C}^E(\mathcal{L})$ for any γ can be ignored as such problems can be solved in $O(1)$ communication rounds by having every node output γ. The exact number of problems that can be pruned as trivial can be measured using the inclusion-exclusion principle as $4 \cdot 2^{30} - 6 \cdot 2^{28} + 4 \cdot 2^{26} - 2^{24}$, reducing the problem space for searching by almost three quarters.

7.2 Transformation-based pruning

Many quaternary LCL problems are not truly unique but transformations of each other in ways that preserve locality. Computing these transformations is faster than algorithm synthesis, so it is desirable to identify problems that are transformations of each other so that algorithm synthesis is only applied on one of them. The locality-preserving transformations for a quaternary edge LCL problem $\mathcal{L}$ are output isomorphism, rotation and mirroring.

A quaternary edge LCL problem $\mathcal{L}'$ is an output isomorphism of $\mathcal{L}$ if there exists a bijection $f : \Gamma \to \Gamma$ such that $(\gamma, \gamma', d) \in \mathcal{C}^E(\mathcal{L})$ if and only if $(f(\gamma), f(\gamma'), d) \in \mathcal{C}^E(\mathcal{L}')$. Locality is preserved, as only the exact output labels used are changed while their relative constraints stay the same. An algorithm A that solves $\mathcal{L}$ in $O(\log^* n)$ communication rounds can trivially solve $\mathcal{L}'$ in $O(1)$ additional time by setting each node's output according to the function f.

A quaternary edge LCL problem $\mathcal{L}'$ is a rotation of $\mathcal{L}$ if $(\gamma, \gamma', d) \in \mathcal{C}^E(\mathcal{L})$ if and only if $(\gamma, \gamma', d') \in \mathcal{C}^E(\mathcal{L}')$ for $d \neq d'$. Locality is preserved for rotations, as an algorithm A that solves $\mathcal{L}$ can be altered to treat northbound edges as eastbound and vice versa, after which it produces the correct output for $\mathcal{L}'$.

A quaternary edge LCL problem $\mathcal{L}'$ is a mirroring of $\mathcal{L}$ if $(\gamma, \gamma', d) \in \mathcal{C}^E(\mathcal{L})$ if and only if $(\gamma', \gamma, d) \in \mathcal{C}^E(\mathcal{L}')$. Locality is preserved for mirrorings, as an algorithm A that solves $\mathcal{L}$ can be altered to treat eastbound edges as westbound, and northbound edges as southbound, and vice versa, after which it solves $\mathcal{L}'$ correctly.

As each of these transformations preserves locality, any combination of them likewise preserves locality. There are 24 permutations of $|\Gamma|$, in addition to which a problem can be mirrored or rotated in one way each, resulting in problem sets of up to $24 \cdot 2 \cdot 2 = 96$ problems that are reachable from each other by applying these transformations. This does not mean that a single application of algorithm synthesis can consistently determine the locality of 96 problems, as it is not quaranteed that the quaternary edge problems derived from transforming any given base problem are all unique.

In order to avoid having to attempt algorithm synthesis for more than one such transformed version of a quaternary edge problem, a canonical version of each problem is defined such that every quaternary edge problem either is or can be converted to exactly one canonical problem by applying the transformations. There are multiple ways to define the set of canonical problems that satisfy the requirements, but the definition used in the practical implementation of this thesis follows.

Let $T(\mathcal{L})$ be the set of all distinct quaternary edge problems that can be reached by applying the locality-preserving transformations to $\mathcal{L}$. Each of

these problems has a distinct set of legal edges. Since there are 32 possible legal edges, there exists a bijection b that assigns each quaternary edge problem a unique 32-bit identifier. Then the canonical problem can be selected as simply the problem $\mathcal{L}' \in T(\mathcal{L})$ with the least value of $b(\mathcal{L}')$ of all.

8 An optimal algorithm for relaxed 3-coloring

The relaxed 3-coloring problem is the LCL problem where, given a graph G, a color assignment $\varphi : V_G \to \{1, 2, 3\}$ is sought such that for each $u \in V_G$, there exists at most one $(u, v) \in E_G$ such that $\varphi(u) = \varphi(v)$. This problem can be contrasted with the 3-coloring problem, where it is required that such (u, v) does not exist at all for any u. The 3-coloring problem is known to be inherently global in grids [1]. There is ample motivation for studying relaxed versions of 3-coloring, as 4-coloring is known to be local in grids: therefore, problems intuitively between 3-coloring and 4-coloring in complexity can prove helpful in understanding the boundary between the two complexity classes.

For the purposes of this thesis, algorithm synthesis was automated[1] for 4-bit LCL problems specifiable in terms of legal edges. Applying the algorithm synthesis process shows that relaxed 3-coloring is solvable in $O(\log^* n)$ communication rounds.

8.1 Constraint encoding

The problem of relaxed 3-coloring is encoded as a four-bit pairwise labeling problem as follows. Each node is assigned four bits. The leading two bits encode the color label of the node, leaving the bit pair 00 unused. The trailing two bits encode a direction in $\{\mathcal{N}, \mathcal{E}, \mathcal{S}, \mathcal{W}\}$. For a node u, let $\varphi(u)$ denote the color part of the label and $d(u)$ denote the direction part. Informally, $d(u)$ indicates a direction in which the node knows it has a neighbor of a matching color in, enabling the restriction that neighbors of matching color

[1] The automated synthesis tools will be made available at https://github.com/Kviiri/lcl-toolkit

do not appear in any other direction while still only checking the endpoints of individual edges. For convenience, denote the opposite direction of the directional label of u as $d^R(u)$, such that

$$d^R(u) = \begin{cases} \mathcal{N} & \text{if } d(u) = \mathcal{S} \\ \mathcal{E} & \text{if } d(u) = \mathcal{W} \\ \mathcal{S} & \text{if } d(u) = \mathcal{N} \\ \mathcal{W} & \text{if } d(u) = \mathcal{E} \end{cases}$$

.

The binary encoding used for the direction is 00 for $\mathcal{N}$, 11 for $\mathcal{S}$, 01 for $\mathcal{E}$ and 10 for $\mathcal{W}$. This means $d^R(u)$ can be obtained simply as the one's complement of $d(u)$.

A 4-bit labeling for a grid G is a legal relaxed 3-coloring encoded in this manner if for every $(u, v) \in E_G$ where $\varphi(u) = \varphi(v)$ and $l = \ell_G(u, v) \in \{N, E\}$, $d(u) = l$ and $d(v) = d^R(u)$. Furthermore, as the representation uses two bits but there are only three legal colors, the extra color is excluded by requiring that $\varphi(u) \neq 0$ for all $u \in V_G$.

Theorem 8. *φ is a valid relaxed 3-coloring for any sufficiently large grid G.*

Proof. Consider a node $u \in V_G$. Cases where $\varphi(u) = 0$ are already excluded, so the only way for φ to not be a legal relaxed 3-coloring is for there to exist $(u, v), (u, w) \in E_G$ for $v \neq w$ such that $\varphi(u) = \varphi(v) = \varphi(w)$.

Consider an edge $(u, v) \in E_G$. As $\varphi(u) = \varphi(v)$, $d(u) = \ell_G(u, v)$ and $d(v) = d^R(u)$ in order not to violate the constraints on the edge (u, v) or (v, u). However, as $\varphi(u) = \varphi(w)$, it must hold that $d(u) = \ell_G(u, w)$ as well. It follows that $\ell_G(u, v) = \ell_G(u, w)$, which in a grid would indicate $v = w$. This cannot be the case for any grid G of dimensions $n \times n$ if $n \geq 3$. It follows that in a sufficiently large grid G, for each $u \in V_G$ there can only be at most one $(u, v) \in E_G$ such that $\varphi(u) = \varphi(v)$. □

The algorithm found is based on the reduction to the computation of MIS introduced in Section 5.3. First, an MIS is computed in $O(\log^* n)$ communication rounds, for example using the method covered in Section 5.2. A

function f mapping k-tiles of dimensions $w \times h$ to $\Gamma = \{1,2,3\} \times \{\mathcal{N}, \mathcal{E}, \mathcal{S}, \mathcal{W}\}$ is then applied in $O(k)$ communication for a constant k for each $u \in V_G$ to determine which tile they are in and set their label accordingly. For formal correctness, each node can truncate the trailing two bits of their output label, resulting in a true 3-coloring without additional output symbols.

Let $k = 1, w = 2, h = 3$. The neighborhood graph for these values consists of the sixteen 1-tiles of dimensions 2×3 listed in Section 6.2 as well as the edges formed by 1-tiles of dimensions 4×2 and 3×3. For conciseness, these tiles shall be referred to as sets of tuples indicating locations of anchors inside the tile. For instance,

the tile $\boxed{\begin{array}{c}10\\01\\10\end{array}}$ will be represented as the set $\{(0,0),(1,1),(0,2)\}$.

With the tile neighborhood graph M, the problem is encoded as a CNFSAT as shown in Section 6.2. As there are four bits per output label, each $u \in V_M$ with $f_V(u) = i$ is assigned four variables: $(x_{4i+1}, x_{4i+2}, x_{4i+3}, x_{4i+4})$. Variables x_{4i+1} and x_{4i+2} encode $\varphi(u)$ while x_{4i+3} and x_{4i+4} encode $d(u)$.

The clauses used to specify the problem for a SAT solver are as follows. For every $(u,v) \in E_M$ with $\ell_M(u,v) \in \{\mathcal{N}, \mathcal{E}\}$, enumerate all ways to assign them a four-bit label that results in an illegal labeling between them. These consist of the 112 ways to set the variables where one or both of $\varphi(u)$ or $\varphi(v)$ is 0, and the 45 ways to set the variables where $\varphi(u) = \varphi(v)$ and each is a legal color, but $d(u)$, $d(v)$ or both are set incorrectly. In total, this creates 157 ways a particular edge can be illegal.

There are, in total, 146 edges in E_M. For each $e \in E_M$ with $\ell_M(e) \in \{\mathcal{N}, \mathcal{E}\}$, there is the opposite edge e^R with $\ell_M(e^R) \in \{\mathcal{S}, \mathcal{W}\}$ whose constraint does not need to be checked. This leaves the number of CNF clauses required at $\frac{146}{2} \cdot 157 = 11461$. Each such clause rules out a particular illegal pairwise labeling of a pair of nodes, and with all illegal labelings ruled out, only locally consistent ones can appear in the ultimate solution. As an example, for $(u,v) \in E_M$ for which $\ell_M(u,v) = \mathcal{N}$, $f_V(u) = i$ and $f_V(v) = j$, the clause

$$(\neg x_{4i+1} \vee \neg x_{4i+2} \vee \neg x_{4i+3} \vee \neg x_{4i+4} \ \vee \ \neg x_{4j+1} \vee \neg x_{4j+2} \vee \neg x_{4j+3} \vee \neg x_{4j+4})$$

rules out the case where $\varphi(u) = \varphi(v) = 3$ and $d(u) = d(v) = \mathcal{N}$, a pairwise labeling that is illegal because $\varphi(u) = \varphi(v)$ but $d(v) \neq \mathcal{S}$.

One should note that in terms of clauses required, this approach does not produce a minimal CNF representation for the constraints of the problem. However, the approach of creating a clause specifying at least one bit of difference from each illegal outcome is versatile and easier to produce for an arbitrary problem than more minimal representations. It is also easy to see the correctness of this approach regardless of the specific problem.

8.2 The output mapping

One automatically derived output mapping that satisfies the constraints of the relaxed 3-coloring problem is presented in the following table:

1-tile u	$\varphi(u)$	$d(u)$
{ (1, 1) }	1	$\mathcal{N}$
{ (1, 2) }	1	$\mathcal{S}$
{ (0, 0), (1, 1) }	1	$\mathcal{S}$
{ (0, 0), (1, 2) }	1	$\mathcal{S}$
{ (0, 0) }	2	$\mathcal{W}$
{ (0, 2) }	2	$\mathcal{N}$
{ (1, 0) }	2	$\mathcal{S}$
{ (0, 0), (0, 2) }	2	$\mathcal{W}$
{ (1, 0), (0, 2) }	2	$\mathcal{E}$
{ (1, 1), (0, 2) }	2	$\mathcal{N}$
{ (0, 0), (1, 1), (0, 2) }	2	$\mathcal{W}$
{ (0, 1) }	3	$\mathcal{E}$
{ (0, 1), (1, 2) }	3	$\mathcal{E}$
{ (1, 0), (0, 1) }	3	$\mathcal{N}$
{ (1, 0), (1, 2) }	3	$\mathcal{W}$
{ (1, 0), (0, 1), (1, 2) }	3	$\mathcal{W}$

One should note that the direction label $d(u)$ is superfluous for nodes with no neighbors of the same color.

The correctness of this labeling can be verified by observing pairs of $(u, v) \in E_M$ with $\ell_M(u, v) \in \{\mathcal{N}, \mathcal{E}\}$ and $\varphi(u) = \varphi(v)$. Such pairs are:

1-tile u	1-tile v	$\ell_M(u, v)$
{ (1, 1) }	{ (1, 2) }	$\mathcal{N}$
{ (1, 1) }	{ (0, 0), (1, 2) }	$\mathcal{N}$
{ (0, 1) }	{ (1, 0), (1, 2) }	$\mathcal{E}$
{ (1, 0), (0, 2) }	{ (0, 0) }	$\mathcal{E}$

For each of them it holds that $d(u) = \ell_M(u, v)$ and $d^R(v) = \ell_M(v, u)$. Therefore, the labeling is legal, and therefore it can be used as the mapping f for the purposes of algorithm synthesis. It follows that a relaxed 3-coloring is computable in $O(\log^* n)$ communication rounds for a grid G of dimensions $n \times n$ as follows. First, label V_G with $\{1, 0\}$ such that nodes labeled 1 form a maximal independent set in G^k. Since $k = 1$, $G^k = G$. This takes $O(\log^* n)$ communication rounds.

Next, each node $u \in V_G$ uses $O(k)$ communication rounds to identify which 1-tile T they belong to due to the assignment of local anchors, and sets their output label as $f(T)$ while truncating the trailing two bits that encode the direction part of the label. This results in a valid relaxed 3-coloring. Since $k = O(1)$ the time complexity of the algorithm is $O(\log^* n)$.

8.3 Example of application

Consider a grid G of dimensions 5×5, labeled with the maximal independent set shown in Figure 7.

Then f is applied such that each node chooses the output based on the tile T they are in posion $(0, 1)$, or left center, of T. For instance, the node in the top-left corner chooses its output based on the tile $\{(0, 1), (1, 2)\}$, resulting in the color 3. The node to its east chooses its output based on the tile $\{(1, 0), (0, 2)\}$ and receives the color 2.

When colored using f, the final assignment of colors is shown in Figure 8.

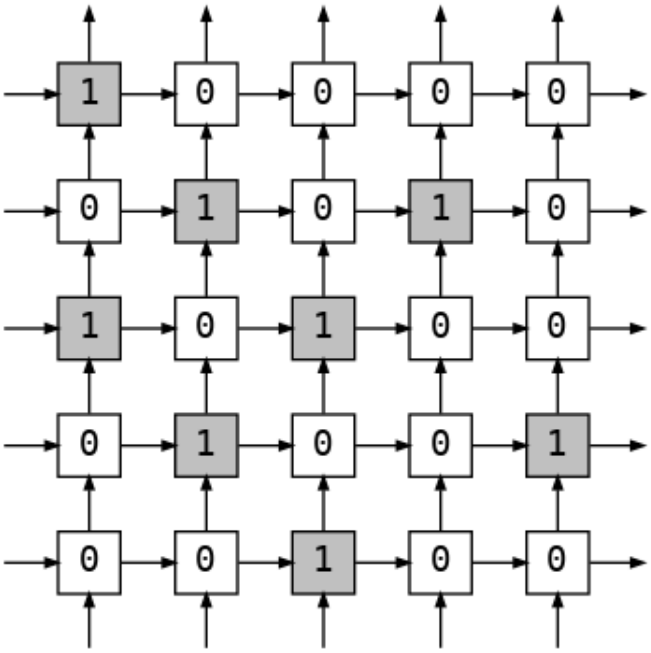

Figure 7: The grid G labeled with a MIS. Nodes labeled 1 and highlighted in grey are members of the MIS.

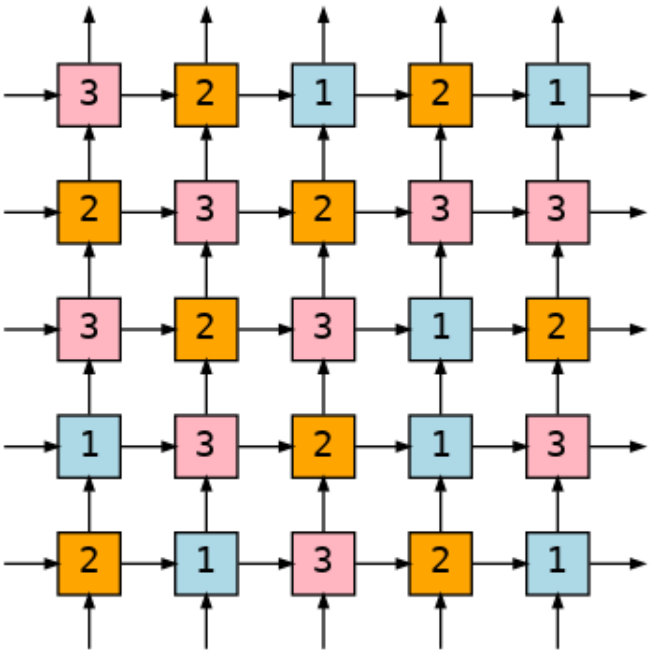

Figure 8: The grid G with a relaxed 3-coloring.

8.4 Semi-relaxed 3-coloring

A *semi-relaxed* 3-coloring problem is an LCL problem where, given a grid G, a color assignment $\varphi : V_G \to \{1, 2, 3\}$ is sought such that for each $(u, v) \in E_G$, if $\varphi(u) = \varphi(v)$ then $\ell_G(u, v) = \mathcal{N}$.

With relaxed 3-coloring known to be local and normal 3-coloring known to be global, the semi-relaxed 3-coloring problem is an interesting target for study. It follows from the constraints of the problem that an algorithm capable of creating a semi-relaxed 3-coloring needs to find true 3-colorings for each directed cycle that runs horizontally across G. The problem of 3-coloring directed cycles was proven by Brandt et al. [1] to be local, so there is no theoretical obstacle to finding an algorithm that computes semi-relaxed 3-coloring in $O(\log^* n)$ communication rounds.

Algorithm synthesis was attempted for semi-relaxed 3-coloring using three sets of parameters: attempts with $k = 1, w = 5, h = 3$ and with $k = 2, w = 5, h = 6$ resulted in no local algorithm being found. However, with $k = 3, w = 7, h = 3$ a local algorithm was found. There are 2079 of 3-tiles of dimensions 7×5, and therefore the mapping required for the algorithm is too large to reproduce here. The mapping, along with the tools used to create it, will be made available in the author's GitHub [10].

References

[1] Brandt, Sebastian *et al.*: LCL problems on grids. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, pages 101–110, New York, NY, USA, 2017. ACM, ISBN 978-1-4503-4992-5. http://doi.acm.org/10.1145/3087801.3087833.

[2] Chang, Yi Jun and Pettie, Seth: *A time hierarchy theorem for the local model.* arXiv preprint arXiv:1704.06297, 2017.

[3] Cole, Richard and Vishkin, Uzi: *Deterministic coin tossing with applications to optimal parallel list ranking.* Information and Control, 70(1):32–53, 1986.

[4] Goldberg, Andrew, Plotkin, Serge, and Shannon, Gregory: *Parallel symmetry-breaking in sparse graphs.* In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 315–324, 1987.

[5] Graham, Ronald L, Rothschild, Bruce L, and Spencer, Joel H: *Ramsey theory*, volume 20. John Wiley & Sons, 1990.

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

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

[8] Panconesi, Alessandro and Rizzi, Romeo: *Some simple distributed algorithms for sparse networks.* Distributed computing, 14(2):97–100, 2001.

[9] Schneider, Johannes and Wattenhofer, Roger: *A log-star distributed maximal independent set algorithm for growth-bounded graphs.* In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing*, pages 35–44, 2008.

[10] Viiri, Kalle: *lcl-toolkit.* `https://github.com/Kviiri/lcl-toolkit`, 2020.