# 1 Definitions

## 1.1 Notation

We will use $\delta(v)$ to refer to the set of edges adjacent to $v$. We identify half-edges in a graph $G = (V, E)$ with a pair $(e, v)$ where $e \in \delta(v)$, referring to the half of $e$ adjacent to $v$.

We use $\{\{\star\}\}$ to refer to multisets, and the following notation for multisubsets of a set $A$:

- $A^{\{\{k\}\}}$ refers to the set of all multisubsets of $A$ of size $k \in \mathbb{N}$,
- $A^{\{\{\mathbb{N}\}\}}$ refers to the set of all finite multisubsets of $A$,
- $A^{\{\{\star\}\}}$ refers to the set of all multisubsets of $A$.

## 1.2 Common and uncommon assumptions

**$T$-hop neighbourhoods:** we define a *$T$-hop centered neighbourhood* as a *centered graph of radius $T$* using the definitions from page 4 of [NS93]. Note that, in the case of subgraphs, the $T$-hop neighbourhood of $v$ may be different from the subgraph induced by all nodes that have distance at most $T$ from $v$.

**Local model:** we work in the *deterministic* LOCAL model with input labels. No assumptions are made about the number of input or output labels. We assume the nodes are aware of the exact number $n$ of nodes in the input graph, but not of any information about the maximum degree $\Delta$. Because of this information, we equivalently describe a LOCAL algorithm as a $T(n)$-round communication algorithm with unbounded message size, or as a *possibly uncomputable* function from $T(n)$-hop centered neighbourhoods to output labels. Additionally, we assume there is a finite integer $c$ such that every node is assigned a unique ID in the set $\{1, \ldots, n^c\}$, but the nodes are not aware of the value of $c$.

**Solvability:** we say that a problem is *weakly solvable* if there are finitely many unsolvable instances for it (this includes problems that are always solvable). We can, and often will, treat weakly solvable problems as solvable problems by implictly adding one output label $U$ and requiring the problem to output $U$ on all nodes if the instance is unsolvable; this requires only constant time, so it does not affect asymptotic complexity. We say that a problem is *strongly unsolvable* if there are infinitely many unsolvable instances for it. We call the maximum diameter of an unsolvable instance the *solvability horizon* of a weakly solvable problem; if the problem is always solvable, we define it as 0, and if it is strongly unsolvable, we define it as $\infty$.

**LCL problems:** we define LCL problems as tuples $\Pi = (\Delta, \Sigma_{\text{in}}, \Sigma_{\text{out}}, r, \mathcal{C})$ where:

- $\Delta$ is a natural number, representing the maximum degree of the graph class considered,
- $\Sigma_{\text{in}}$ is a set of input labels,
- $\Sigma_{\text{out}}$ is a set of output labels,
- $r$ is a finite integer, called the *radius* of $\Pi$, and
- $\mathcal{C}$ is a *finite* set of $r$-hop centered neighbourhoods of maximum degree $\Delta$, where each node is labelled by a pair in $\Sigma_{\text{in}} \times \Sigma_{\text{out}}$.

Though we haven't explicitly stated this in the definition, WLOG we can and will also assume that $\Sigma_{\text{in}}$ is finite (else the problem would be trivially strongly unsolvable, since only finitely many input labels appear in $\mathcal{C}$) and that $\Sigma_{\text{out}}$ is finite (any labels that don't appear in $\mathcal{C}$ will not appear in any valid solution and can be ignored).

**Node-edge checkable problems:** we define node-edge checkable problems as tuples $\Pi = (\Delta, \Sigma_{\text{in}}, \Sigma_{\text{out}}, \mathcal{N}, \mathcal{E})$ where:

- $\Delta$ is a natural number, representing the maximum degree of the graph class considered,
- $\Sigma_{\text{in}}$ is a set of input labels,
- $\Sigma_{\text{out}}$ is a set of output labels,
- $r$ is a finite integer, called the *radius* of $\Pi$, and
- $\mathcal{N}$ is an indexed set $\{\mathcal{N}_{d,\chi}\}_{d\leq\Delta,\chi\in\Sigma_{\text{in}}}$ where each $\mathcal{N}_{d,\chi}$ is a subset of $\Sigma_{\text{out}}^{\{\{d\}\}}$ (representing the set of allowed configurations for a node of degree $d$ with input label $\chi$),
- $\mathcal{E}$ is a subset of $\Sigma_{\text{out}}^{\{\{2\}\}}$ (representing the set of allowed configurations over edges).

Similar reasoning to the above as to why we can implictly assume that $\Sigma_{\text{in}}$ and $\Sigma_{\text{out}}$ are finite sets.

## 1.3 New-ish definitions

**Hanging trees:** *connected* tree graphs containing a finite number of incomplete half-edges, called *hooks*. Specifically, we will call hanging trees with one hook *ornaments* and hanging trees with two hooks *tinsels*. We call nodes adjacent to one or more hooks *hinge nodes*.

**Hanging tree sets:** we define as $\mathcal{T}_{k,\Delta}$ the set of all hanging trees with $k$ hooks and maximum degree $\Delta$. Additionally, we define $\mathcal{T}_k = \bigcup_{\Delta\in\mathbb{N}} \mathcal{T}_{k,\Delta}$ as the set of all hanging trees with $k$ hooks. Note that both are countably infinite sets.

**Class:** given an ornament $T$ and a node-edge checkable problem or FSL problem $\Pi$, we define the *class* of $T$ to be the set $\mathfrak{Cl}(T) \subseteq \Sigma_{\text{out}}$ containing exactly the labellings of the hook that can be completed to a valid labelling for $T$ according to $\Pi$. Up to the changed role of nodes and edges, this is the *class* definition from Section 3 of [CP17]. Note that since $\Sigma_{\text{out}}$ is finite, so is the set of all possible classes.

## 1.4 Actually new definitions

**Finitely Represented Configuration:** given an alphabet of symbols $\Sigma$, we call a *finitely represented configuration* of $\Sigma$ a pair of a *finite* multisubset of $\Sigma$ (the *requirement*) and a *finite* subset of $\Sigma$ (the *filler*). We denote the set of finitely represented configurations in $\Sigma$ as

$$\Sigma^{FSL} := \Sigma^{\{\{\mathbb{N}\}\}} \times [\Sigma]^{<\omega}$$

**Extended Finite State Labelling problems:** we define Extended FSL problems as tuples $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, f, \mathcal{N}, \mathcal{E})$ where:

- $\Sigma_{\text{in}}$ is a *finite* set of input labels,
- $\Sigma_{\text{out}}$ is a *finite* set of output labels,
- $f : \mathbb{N} \to 2^{\Sigma_{\text{in}}}$ is a *computable* function describing which input labels are allowed to appear for a node of degree $d \in \mathbb{N}$ which is surjective on some cover of $\Sigma_{\text{in}}$ (in other words, each label in $\Sigma_{\text{in}}$ appears in some set in $f(\mathbb{N})$),
- $\mathcal{N} : \Sigma_{\text{in}} \to 2^{(\Sigma_{\text{out}})^{FSL}}$ is a function assigning to each input label a set of *finitely represented configurations* of $\Sigma_{\text{out}}$, and

- $\mathcal{E}$ is a set of pairs of elements of $\Sigma_{\text{out}}$.

An *instance* for $\Pi$ is a graph $G = (V, E)$ together with a labelling function $l : V \to \Sigma_{\text{in}}$ such that $l(v) \in f(\deg(v))$, that is, the labelling is coherent with the "allowed" labels for each degree. A *solution* for this instance is a labelling $s$ of the half-edges of $G$ such that:

- $\forall v \in V$, let $S(v) := \{\{s(e, v) \,|\, e \in \delta(v)\}\}$; then there exists $(R, F) \in \mathcal{N}(l(v))$ such that $R \subseteq S(v)$ and $(S(v) \setminus R) \subseteq F^{\{\{\star\}\}}$, that is to say, every symbol in the requirement appears exactly once in $S(v)$, and every other symbol of $S(v)$ is a filler symbol, and
- $\forall e = \{u, v\} \in E$ we have $\{s(e, u), s(e, v)\} \in \mathcal{E}$.

**Finite State Labelling problems:** we define FSL problems as tuples $\Pi = (\Sigma_{\text{out}}, \mathcal{N}, \mathcal{E})$ where:

- $\Sigma_{\text{out}}$ is a *finite* set of output labels,
- $\mathcal{N} \subseteq 2^{(\Sigma_{\text{out}})^{FSL}}$ is a *finite* set of finitely represented configurations of $\Sigma_{\text{out}}$,
- $\mathcal{E}$ is a set of pairs of elements of $\Sigma_{\text{out}}$.

An *instance* for $\Pi$ is any graph; a solution for this instance is a labelling $s$ of the half-edges of $G$ such that:

- $\forall v \in V$, let $S(v) := \{\{s(e, v) \,|\, e \in \delta(v)\}\}$; then there exists $(R, F) \in \mathcal{N}$ such that $R \subseteq S(v)$ and $(S(v) \setminus R) \subseteq F^{\{\{\star\}\}}$, that is to say, every symbol in the requirement appears exactly once in $S(v)$, and every other symbol of $S(v)$ is a filler symbol, and
- $\forall e = \{u, v\} \in E$ we have $\{\{s(e, u), s(e, v)\}\} \in \mathcal{E}$.

**Minimum degree of a label:** we define the *minimum degree* of an input label $\chi \in \Sigma_{\text{in}}$ as $\deg_{\min}(\chi) := \min\{d \in \mathbb{N} \,|\, \chi \in f(d)\}$ (the minimum degree for which that input label is allowed). This set is never empty. Generally, we can assume that $\forall (R, F) \in \mathcal{N}(\chi)$ we have $|R| = \deg_{\min}(\chi)$, since a configuration with a longer requirement would be unattainable by a finite number of degrees (which can be encoded by putting them in separate input classes) and any configuration with a shorter requirement implicitly requires using a number of filler symbols, which can be encoded by adding all combinations of filler symbols to the requirements multiset.

**Width:** we define the *width* of a EFSL problem $\Pi$ as the maximum length of one of its requirements: accounting for hidden requirements encoded in the degree, we get

$$\max\left\{\max_{\chi \in \Sigma_{\text{in}}} \deg_{\min}(\chi), \max\{|R| \,|\, \exists \chi \in \Sigma_{\text{in}}, \exists F \subset \Sigma_{\text{out}} : (R, F) \in \mathcal{N}(\chi)\}\right\}$$

For FSL problems we define it as $\max\{|R| \,|\, \exists F (\subset \Sigma_{\text{out}} : R, F) \in \mathcal{N}\}$.

**Height:** we define the *height* of an EFSL problem as $h(\Pi) := \max\{|\mathcal{N}(\chi)| \,|\, \chi \in \Sigma_{\text{in}}\}$ (the maximum number of configurations any node can be in). For FSL problems we define it as $|\mathcal{N}|$.

**Restricted (E)FSL:** we define the *restriction* of an (E)FSL the *node-edge checkable* problem with input labels obtained by replacing every finitely represented configuration $(R, F)$ with the configurations $\{R \cup \{\{x\}\} \,|\, x \in F\}$.

# 2 List of results

1. Extended FSL = FSL (up to changing the number of labels) input/degree info in the structure and output labels

2. Node-edge checkable problems $\subseteq$ FSL
3. Weak and strong solvability are decidable on FSL

# 3 List of possible results

1. If a FSL problem is known to be $\Omega(\log n)$ on trees, its complexity on trees is decidable and an efficient algorithm can be found. (This holds because of Gustav's proof which should be stronger, it's in here because I still haven't written out all the details of the FSL based one)
2. (Belief I haven't been able to fully prove) For each FSL problem $\Pi$ there is a value $d$ that only depends on the problem description (specifically, number of labels, height and width) such that there is a worst-case family of graphs for $\Pi$ of maximum degree $d$. My theory is either width+1 or width·height (also possible: width+height, using one of each filler). Latter is required for solvability.
3. (Investigating) A form of the Round Elimination procedure applies to FSL problems.

# 4 Proofs

## 4.1 Old(er) proofs that I finally formalised

### 4.1.1 Solvability

We include definitions of B-hypergraph, B-hyperpath and hyperpath tree based on [TT09] and [AGN97, Section 2]

**Definition 4.1.** [TT09, Definition 2.2] A *directed hypergraph* $H$ is a pair $(V, E)$, where V is a finite set and $E \subseteq 2^V \times 2^V$ such that, for every $e = (T(e), H(e)) \in E, T(e) \neq \emptyset, H(e) \neq \emptyset$, and $T(e) \cap H(e) = \emptyset$. A B-hyperedge is a hyperedge e = (T (e), H(e)) such that $|H(e)| = 1$; a directed B-hypergraph is a directed hypergraph H such that each hyperedge in H is a B-hyperedge.

**Definition 4.2.** [AGN97, Definition 2.4] Let $H = (V, E)$ be a directed B-hypergraph, let $X \subseteq V$ be a non-empty subset of nodes and $t$ be a node in $V$. There is a *B-hyperpath* from $X$ to $t$ in $V$ if $y \in X$, or there is an hyperedge $e = (Z, \{y\})$ such that there is a B-hyperpath from $X$ to each $z_i \in Z$.

**Definition 4.3.** [AGN97, Definition 2.5] Let $H = (V, E)$ be a directed B-hypergraph, let $X \subseteq V$ be a non-empty subset of nodes and $t$ be a node in $V$ such that there is a B-hyperpath from $X$ to $t$. A *hyperpath tree* is a tree $T_{X,t}$ defined as follows:

- if $t \in X$, $T_{X,t}$ is the empty tree,
- if there is a hyperarc $e = (Z, t)$ and hyperpaths from $X$ to each $z_i \in Z$, then $T_{X,t}$ consists of a root labelled $e$ having as subtrees the hyperpath trees $T_{X,z_i}$ for each $z_i \in Z$.

**Theorem 4.1.** *Let $\Pi$ be a node-edge checkable problem. Then it is possible to decide whether $\Pi$ is strongly unsolvable or weakly solvable on trees and compute the solvability horizon.*

*Proof.* Let $\Pi = (\Delta, \Sigma_{\text{in}}, \Sigma_{\text{out}}, \mathcal{N}, \mathcal{E})$, we construct the *existential graph of classes* of $\Pi$, which is a directed B-hypergraph we denote with $H_\Pi^{Cl}$ and define as follows:

- the set of nodes of $H_\Pi^{Cl}$ is $\mathcal{P}(\Sigma_{\text{out}})$ with an extra node labelled $\perp$,
- there is an edge from $\{Cl_1, \dots, Cl_k\}$ to $Cl$ if and only if there is a hanging tree of class $Cl$ whose set of sub-hanging trees (meaning, the hanging trees obtained by removing the hook node and all of its adjacent half-edges) is exactly $\{Cl_1, \dots, Cl_k\}$ (type A),
- there is an edge from $\{Cl_1, Cl_2\}$ to $\perp$ if and only if $(Cl_1 \times Cl_2) \cap \mathcal{E} = \emptyset$[1] (type B),
- there are no other edges.

Additionally, we mark the nodes corresponding to $\{\mathcal{N}_{1,\chi}\}_{\chi \in \Sigma_{\text{in}}}$ as "starting" nodes.

The proof then roughly goes as follows: we show that an algorithm exists that builds $H_\Pi^{Cl}$ in finite time [★1], we show a relation between its longest hyperpath trees and the solvability horizon of $\Pi$ [★2], and finally show we can compute the longest hyperpath trees in finite time [★3].

[★1] First, we initialise the graph with the node set and the edges of type (B), which can be done in time $O(2^{|\Sigma_{\text{out}}|})$ by iterating over the powerset. To find the edges of type (A) we define a set of *reached classes* which we initialise as the set of starting nodes $\mathcal{C} := \{\mathcal{N}_{1,\chi}\}_{\chi \in \Sigma_{\text{in}}}$; if $\emptyset \in \mathcal{C}$, the problem is strongly unsolvable and the solvability horizon is $\infty$. We assume this is not the case.

Then, we iterate over all the ornaments of height 1 (ie containing only the hook node and leaves) and max degree $\Delta$, over all elements of $\Sigma_{\text{in}}$ as input labels to the hook node, and over all combinations of elements of $\mathcal{C}$ over the leaves. Note that there are finitely many such labelled trees.

Now, we compute the class of each ornament as if it were an instance of $\Pi$ on the hook node, with the leaves accepting only and all of the labels in their input label. Every time we do this, we add the class to a set $\mathcal{U}$ and add an edge in $H_\Pi^{Cl}$ from the labels that appear on to the node corresponding to the class; if this edge already exists, we can skip this step. For convenience, if we are adding a new edge, we can store the tree we were considering as a label to it.

At the end of this iteration, we check whether $\mathcal{U} \subseteq \mathcal{C}$; in this case, we stop. If $\mathcal{U} \nsubseteq \mathcal{C}$ we repeat the process by setting $\mathcal{C}' := \mathcal{C} \cup \mathcal{U}$. This means $|\mathcal{C}'| \geq |\mathcal{C}| + 1$; since $\mathcal{C}', \mathcal{C} \subseteq \mathcal{P}(\Sigma_{\text{out}})$ the process has to terminate in at most $2^{2^{|\Sigma_{\text{out}}|}}$ repetitions.

To show that this algorithm is correct, we show that every possible ornament is encoded in the edges of type (A) by induction on the height $h$ (the maximum distance between the hook node and a leaf). This is vacuously true for $h = 0$, meaning single leaves which do not have any sub-hanging tree; consider instead an ornament $H$ of height $h + 1$. If every one of its sub-hanging trees (of height $\leq h$) was previously encoded, their classes were added $\mathcal{C}$ at some step. In the next step, we must have built an ornament of height 1 with each leaf corresponding to one of these sub-trees and added an edge encoding $H$ if it wasn't already present. Then $H$ is encoded, and by induction, so are all finite ornaments; we denote as $f$ the function mapping ornaments to their encoding edges.

[★2] We claim that the unsolvability horizon of $\Pi$ corresponds to the maximum over the diameters of all hyperpath trees starting from the set $S$ of starting/leaf nodes and ending in $\perp$. More precisely, we claim for each such *finite* hyperpath tree of diameter $d$ we can construct an unsolvable instance of diameter $d+1$ and for each unsolvable instance of diameter $d+1$ we can construct a hyperpath tree of diameter $d$.

[1] with a slight abuse of notation, we consider $Cl_1 \times Cl_2$ to be a multiset rather than an ordered pair.

Let $T_{S,\perp}$ be a hyperpath tree from $S$ to $\perp$. Since $\perp \notin S$, there is a set $\{Cl_1, Cl_2\}$ such that:

- there are hyperpath trees from $S$ to $Cl_1, Cl_2$, and
- $(Cl_1 \times Cl_2) \cap \mathcal{E} = \emptyset$.

We traverse the hyperpath tree from $S$ to $Cl_1$; every time we traverse an hyperedge $h = (T, Cl)$, we replace each node corresponding to the class $Cl$ in the previous ornament with the ornament we labelled $h$ with. The result is an ornament $G_1$ of class $Cl_1$, of diameter equal to the hypertree depth $+1$; we repeat this procedure to get an ornament $G_2$ of class $Cl_2$. Then we join the two hooks of $G_1, G_2$ into a single edge $e$ to get a tree $T$; since $(Cl_1 \times Cl_2) \cap \mathcal{E} = \emptyset$ there is no valid labelling for $e$ and $T$ is an unsolvable instance of diameter equal to $\text{depth}(G_1) + \text{depth}(G_2) + 1 = \text{diam}(T_{S,\perp}) - 1$.

Conversely, let $G$ be an unsolvable instance for $\Pi$ and let $e$ be any edge of $G$. Let $G_1, G_2$ be the two ornaments obtained by splitting $e$ in half; since no label assignment to $e$ would result in a solution, we have that $(\mathfrak{Cl}(G_1) \times \mathfrak{Cl}(G_2)) \cap \mathcal{E} = \emptyset$ and there is a hyperedge from $\perp$ to $\{\mathfrak{Cl}(G_1), \mathfrak{Cl}(G_2)\}$. From there, we can use the function $f$ described above to map $G_1$ to its encoding edge, and so on recursively to find a hyperpath from the set of starting nodes to $\mathfrak{Cl}(G_1)$ of length equal to $\text{depth}(G_1) - 1$. By repeating this procedure on $G_2$ we get a hyperpath from $S$ to $\perp$ corresponding to a hyperpath tree of diameter $\text{depth}(G_1) - 1 + \text{depth}(G_2) - 1 + 1 = \text{diam}(G) + 1$.

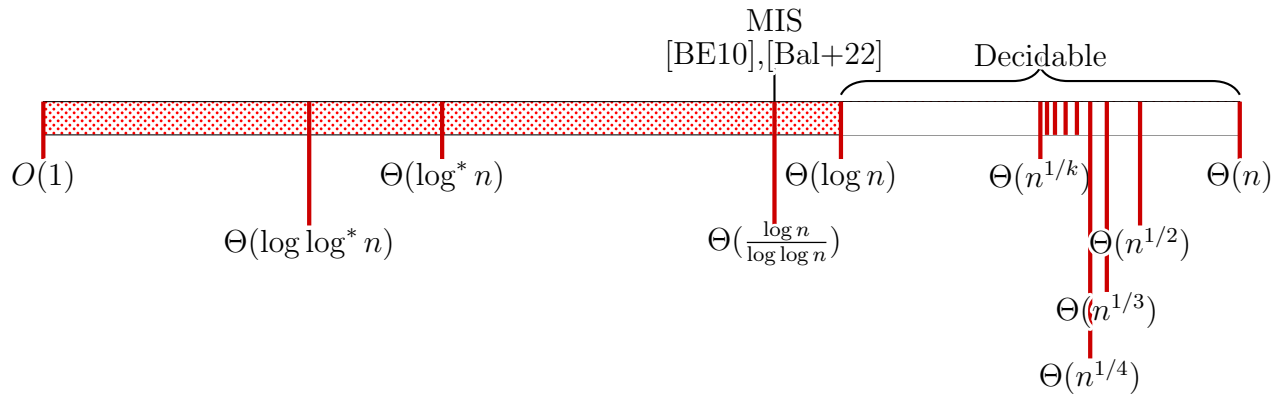
Finally, note that the proof above applies to finite hyperpath trees. If any hyperpath from $S$ to $\perp$ contains a cycle, then we can find hyperpath trees of arbitrarily large finite depth; then we can find unsolvable instances of arbitrarily large depth, and $\Pi$ is strongly unsolvable (with a solvability horizon of $\infty$). Similarly, if no hyperpaths exist between $S$ and $\perp$, then no unsolvable instances exist, and the solvability horizon of $\Pi$ is zero. If at least one path exists and no paths contain cycles, there are finitely many paths; we can iterate over all of them to obtain their maximum diameter and the solvability horizon.[2] □

**Theorem 4.2.** *Let $\Pi$ be a FSL problem. Then it is possible to decide whether $\Pi$ is strongly unsolvable or weakly solvable and compute the solvability horizon (if it exists).*

[2] We are only concerned about finiteness here; a discussion of the complexity of finding the longest hyperpath can be found in [TT09]

# 5 What do we know about the complexity landscape?

## 5.1 Trees




## 5.2 General graphs

# Bibliography


[NS93] Moni Naor and Larry Stockmeyer. "What can be computed locally?" In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing*. STOC '93. New York, NY, USA: Association for Computing Machinery, June 1, 1993, pp. 184–193. ISBN: 978-0-89791-591-5. DOI: 10.1145/167088.167149. URL: https://www.wisdom.weizmann.ac.il/~naor/PAPERS/lcl.pdf (visited on 08/11/2025).

[AGN97] Giorgio Ausiello, Roberto Giaccio, and Umberto Nanni. "Optimal Traversal of Directed Hypergraphs". In: (Nov. 24, 1997). URL: https://www.researchgate.net/publication/2455024_Optimal_Traversal_of_Directed_Hypergraphs.

[TT09] Mayur Thakur and Rahul Tripathi. "Linear Connectivity Problems in Directed Hypergraphs". In: *Theoretical Computer Science* 410.27 (June 28, 2009), pp. 2592–2618. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2009.02.038.

[BE10] Leonid Barenboim and Michael Elkin. "Sublogarithmic Distributed MIS Algorithm for Sparse Graphs Using Nash-Williams Decomposition". In: *Distributed Computing* 22.5 (Aug. 1, 2010), pp. 363–379. ISSN: 1432-0452. DOI: 10.1007/s00446-009-0088-2.

[CP17] Yi-Jun Chang and Seth Pettie. "A Time Hierarchy Theorem for the LOCAL Model". In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS). Berkeley, CA, Oct. 2017, pp. 156–167. ISBN: 978-1-5386-3464-6. DOI: 10.1109/FOCS.2017.23. arXiv: 1704.06297 [cs.DC].

[Bal+22] Alkida Balliu et al. "Distributed $\Delta$-Coloring Plays Hide-and-Seek". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2022. New York, NY, USA: Association for Computing Machinery, June 10, 2022, pp. 464–477. ISBN: 978-1-4503-9264-8. DOI: 10.1145/3519935.3520027.