


Brief Announcement: Distributed Graph Problems Through an Automata-Theoretic Lens

Yi-Jun Chang 

ETH Zürich, Switzerland
yi-jun.chang@eth-its.ethz.ch

Jan Studený 

Aalto University, Finland
jan.studený@aalto.fi

Jukka Suomela 

Aalto University, Finland
jukka.suomela@aalto.fi

Abstract

We study the following algorithm synthesis question: given the description of a locally checkable graph problem Π for paths or cycles, determine in which instances Π is *solvable*, determine what is the *locality* of Π , and construct an asymptotically optimal *distributed algorithm* for solving Π (in the usual LOCAL model of distributed computing). To answer such questions, we represent Π as a nondeterministic finite automaton \mathcal{M} over a unary alphabet, and identify polynomial-time-computable properties of automaton \mathcal{M} that capture the locality and solvability of problem Π .

2012 ACM Subject Classification Theory of computation \rightarrow Formal languages and automata theory; Theory of computation \rightarrow Models of computation; Theory of computation \rightarrow Distributed algorithms

Keywords and phrases Algorithm synthesis, locally checkable labeling problems, LOCAL model, locality, distributed computational complexity, nondeterministic finite automata

Digital Object Identifier 10.4230/LIPIcs.DISC.2020.41

Related Version A full version of the paper is available at <https://arxiv.org/abs/2002.07659>.

1 Introduction

Key Questions: Solvability and Locality. When we encounter a new graph problem Π that we would like to solve in a distributed or parallel setting, there are three basic questions we would like to answer:

1. Is Π always *solvable*, or at least solvable in all but finitely many counterexamples?
2. Is Π solvable *locally*: if I am a node in the middle of a large graph, can I choose my own part of the solution based on the information in my own local neighborhood?
3. If Π is locally solvable, how do we find an efficient *algorithm* for solving it?

Locality is a powerful property: in many setting a problem can be solved efficiently if and only if it is highly local. For the sake of concreteness, in this work we will discuss deterministic distributed algorithms in the usual LOCAL model of computing, in which efficient solvability is equivalent to locality, but we emphasize that our work has direct implications also in other models of distributed computing (e.g. CONGEST) and also in models of parallel computing (e.g. PRAM and MPC), especially for the upper bounds.

Focus and Prior Work. We will focus on LCL problems. These are graph problems in which solutions are labelings of nodes and/or edges that can be *verified locally*: if a solution looks feasible in all constant-radius neighborhoods, then it is also globally feasible [4]. Simple examples of LCL problem are finding a proper 3-coloring and finding maximal independent set of a given graph.



© Yi-Jun Chang, Jan Studený, and Jukka Suomela;
licensed under Creative Commons License CC-BY

34th International Symposium on Distributed Computing (DISC 2020).

Editor: Hagit Attiya; Article No. 41; pp. 41:1–41:3



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Given the description of an LCL problem, ideally we would like to understand its solvability and locality *in a fully automatic fashion*, and *synthesize* efficient distributed and parallel algorithms for the problem, in the broadest possible setting. Unfortunately, in the case of general graphs this is an undecidable problem [3, 4], and even if we only consider labeled paths and cycles, the task is known to be at least PSPACE-hard [1]. On the other hand, the distributed complexity of LCLs on *unlabeled directed cycles* (consistently oriented cycles) has a simple graph-theoretic characterization [3]. In this work we seek to find the broadest possible setting in which solvability and locality can be decided efficiently.

Contributions. We show how to *automatically answer questions related to both solvability and locality* of any given LCL problem both in directed cycles and paths and in undirected cycles and paths. We show that all such questions are not only decidable but they are in NP or co-NP, and almost all such questions are in P, with the exception of a couple of specific questions that are NP-complete or co-NP-complete. We give a uniform automata-theoretic formalism that enables us to study such questions, leveraging prior work on automata theory.

2 Automata Representation

Node-Edge-Checkable Formalism. There are many equivalent ways to represent LCL problems (e.g., listing all feasible constant-radius neighborhoods [4]), but the following formalism [2] is convenient for us: we interpret each edge as a pair of *ports*, and the task is to label the ports, subject to constraints on nodes and edges. Note that one can easily use port labels and node and edge constraints to encode e.g. vertex coloring and edge coloring, and indeed the formalism is expressive enough to encode *any* LCL problem on paths and cycles, modulo local preprocessing and postprocessing.

Formally, an LCL problem Π in the node-edge-checkable formalism is a tuple $\Pi = (\Gamma, \mathcal{C}_{\text{edge}}, \mathcal{C}_{\text{node}}, \mathcal{C}_{\text{start}}, \mathcal{C}_{\text{end}})$ consisting of a finite set Γ of output labels, an edge constraint $\mathcal{C}_{\text{edge}} \subseteq \Gamma \times \Gamma$ (that captures feasible labelings of edges), a node constraint $\mathcal{C}_{\text{node}} \subseteq \Gamma \times \Gamma$ (that captures feasible labelings of the internal nodes of cycles and paths), and start and end constraints $\mathcal{C}_{\text{start}} \subseteq \Gamma$ and $\mathcal{C}_{\text{end}} \subseteq \Gamma$ (that capture feasible labelings at the endpoints of paths). We say that Π is *symmetric* if $\mathcal{C}_{\text{edge}}$ and $\mathcal{C}_{\text{node}}$ are symmetric relations and $\mathcal{C}_{\text{start}} = \mathcal{C}_{\text{end}}$; such a problem is well-defined also for undirected paths and cycles.

Automata Representation. We will represent a node-edge-checkable problem Π as a *nondeterministic finite automaton* \mathcal{M}_{Π} over unary alphabet $\Sigma = \{o\}$. We identify the states of an automaton with the edge constraints and the transitions with the node constraints. More precisely, the set of states is $\mathcal{C}_{\text{edge}}$, there is a transition from state ab to state cd whenever $bc \in \mathcal{C}_{\text{node}}$, state $ab \in \mathcal{C}_{\text{edge}}$ is a starting state whenever $a \in \mathcal{C}_{\text{start}}$, and state $ab \in \mathcal{C}_{\text{edge}}$ is an accepting state whenever $b \in \mathcal{C}_{\text{end}}$. Note that there can be multiple starting states; the automaton can choose the starting state nondeterministically.

Classification of States. We classify the states as follows; we say that state $ab \in \mathcal{C}_{\text{edge}}$ is:

- *Repeatable* if there is a walk $ab \rightsquigarrow ab$ in \mathcal{M}_{Π} .
- *Flexible* if for all sufficiently large k there is a walk $ab \rightsquigarrow ab$ of length exactly k in \mathcal{M}_{Π} .
- *Mirror-flexible* if for all sufficiently large k there are walks $ab \rightsquigarrow ab$, $ab \rightsquigarrow ba$, $ba \rightsquigarrow ab$, and $ba \rightsquigarrow ba$ of length exactly k in \mathcal{M}_{Π} .
- A *loop* if there is a state transition $ab \rightarrow ab$ in \mathcal{M}_{Π} .
- A *mirror-flexible loop* if ab is both a mirror-flexible state and a loop.

■ **Table 1** Classification of LCL problems in cycles and paths into types A–K, and the implied characteristics (solvability and locality) for each type.

Type definition:	A	B	C/D	E	F/G	H/I	J/K
· symmetric problem	yes	yes	yes/no	yes	yes/no	yes/no	yes/no
· repeatable state	yes	yes	yes	yes	yes	yes	no
· flexible state	yes	yes	yes	yes	yes	no	no
· loop	yes	yes	yes	no	no	no	no
· mirror-flexible state	yes	yes	no	yes	no	no	no
· mirror-flexible loop	yes	no	no	no	no	no	no
Number of instances:							
· solvable cycles	∞	∞	∞	∞	∞	∞	0
· solvable paths	∞	∞	∞	∞	∞	∞	$< \infty$
· unsolvable cycles	0	0	0	$< \infty$	$< \infty$	∞	∞
· unsolvable paths	$< \infty$	$< \infty$	$< \infty$	$< \infty$	$< \infty$	hard	∞
Locality:							
· directed cycles	$O(1)$	$O(1)$	$O(1)$	$\Theta(\log^* n)$	$\Theta(\log^* n)$	$\Theta(n)$	—
· directed paths	$O(1)$	$O(1)$	$O(1)$	$\Theta(\log^* n)$	$\Theta(\log^* n)$	$\Theta(n)$	$O(1)$
· undirected cycles	$O(1)$	$\Theta(\log^* n)$	$\Theta(n)^\dagger$	$\Theta(\log^* n)$	$\Theta(n)^\dagger$	$\Theta(n)^\dagger$	—
· undirected paths	$O(1)$	$\Theta(\log^* n)$	$\Theta(n)^\dagger$	$\Theta(\log^* n)$	$\Theta(n)^\dagger$	$\Theta(n)^\dagger$	$O(1)^\dagger$

† = assuming a symmetric problem (otherwise it is not well-defined on undirected cycles and paths)

3 Results

Our main result is the classification presented in Table 1. Given any LCL problem Π , we can first construct the automaton \mathcal{M}_Π , and determine if the problem is symmetric and if it contains loops or repeatable, flexible, or mirror-flexible states. Based on these properties, we can then use the table to classify the problem into types A–K. In the full version of this work, we show that we can *determine the type of any given problem in polynomial time*.

Now once we know the type of the problem, we can use Table 1 to directly answer all questions related to solvability and locality of Π in both undirected and directed cycles and paths, with only one exception: the only case which cannot be efficiently answered is deciding the number of unsolvable instances for problems of types H–I; we show that this question is NP-complete. In the full version, we prove that the classification is correct, and we show also how our work connects to prior work on the existence of *synchronizing words* for nondeterministic automata.

References

- 1 Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. The Distributed Complexity of Locally Checkable Problems on Paths is Decidable. In *Proc. PODC*, 2019. doi:10.1145/3293611.3331606.
- 2 Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In *Proc. DISC*, 2020.
- 3 Sebastian Brandt, Juho Hirvonen, Janne H Korhonen, Tuomo Lempiäinen, Patric R J Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznański. LCL problems on grids. In *Proc. PODC*, 2017. doi:10.1145/3087801.3087833.
- 4 Moni Naor and Larry Stockmeyer. What Can be Computed Locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.