Lecture Notes in Computer Science 10641

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

More information about this series at http://www.springer.com/series/7407

Shantanu Das · Sebastien Tixeuil (Eds.)

Structural Information and Communication Complexity

24th International Colloquium, SIROCCO 2017
Porquerolles, France, June 19–22, 2017
Revised Selected Papers

Springer

Editors
Shantanu Das
LIF
Aix-Marseille University
Marseille Cedex 9
France

Sebastien Tixeuil
LIP6
Université Pierre et Marie Curie - Paris 6
Paris
France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-72049-4 ISBN 978-3-319-72050-0 (eBook)
https://doi.org/10.1007/978-3-319-72050-0

Library of Congress Control Number: 2017960878

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at the 24th International Colloquium on Structural Information and Communication Complexity: SIROCCO 2017, held during June 19–21, 2017, at Porquerolles, France. SIROCCO is devoted to the study of the interplay between structural knowledge, communication, and computing in decentralized systems of multiple communicating entities. Special emphasis is given to innovative approaches leading to better understanding of the relationship between computing and communication. The typical areas of interest include distributed computing, communication networks, game theory, parallel computing, social networks, mobile computing (including autonomous robots), peer-to-peer systems, and communication complexity.

This year we received 41 submissions in response to the call for papers and the Program Committee decided to accept 21 papers after a careful review and in-depth discussions. Each submission was reviewed by at least three reviewers; we had a total of 20 Program Committee members supported by 42 additional reviewers. The article "Leader Election in SINR Model with Arbitrary Power Control," by Magnus M. Halldorsson, Stephan Holzer, and Evangelia Anna Markatou received the Best Paper Award at SIROCCO 2017. Selected papers from the colloquium were invited to a special issue of the *Theoretical Computer Science* journal.

In addition to the regular contributed talks, the conference program included invited talks by Faith Ellen, Christian Scheideler, and Christoph Lenzen, and the award lecture by Shmuel Zaks, who received the 2017 SIROCCO Prize for Innovation in Distributed Computing. Short abstracts of all invited lectures, as well as a laudatio summarizing the numerous important innovative contributions of the award recipient Shmuel Zaks, are included in this volume.

We would like to thank all authors for their high-quality submissions and all speakers for their excellent talks at the conference. We are specially grateful to the Program Committee members and all additional reviewers who worked on a tight schedule to deliver an excellent conference program. The preparation of this event was guided by the SIROCCO Steering Committee, headed by Andrzej Pelc; we thank them for their help and support. The local organization of the conference was possible due to the efforts of the members of the Organizing Committee who were supported by the staff from the Laboratoire d'Informatique Fondamentale (LIF) at the Aix-Marseille University. The organizers of SIROCCO 2017 would like to acknowledge the financial support of our sponsors: LIF—Laboratory of Theoretical Computer Science, LabEx-Archimede Center of excellence, FRIIAM—Federation for research in Computer Science and Interactions, CNRS France, and Aix-Marseille University. Springer

not only helped with the publication of these proceedings but also sponsored the best paper award. The EasyChair system was used to handle the submission of papers, manage the review process, and generate these proceedings.

August 2017

Shantanu Das
Sebastien Tixeuil

Organization

Program Committee

Dan Alistarh	ETH Zurich, Switzerland
Silvia Bonomi	Sapienza Università di Roma, Italy
Shantanu Das	Aix-Marseille University, France
Yann Disser	TU Darmstadt, Germany
Guy Even	Tel Aviv University, Israel
Antonio Fernandez Anta	IMDEA Networks Institute, Spain
Leszek Gasiniec	University of Liverpool, UK
Rastislav Kralovic	Comenius University, Slovakia
Danny Krizanc	Wesleyan University, USA
Bernard Mans	Macquarie University, Australia
Euripides Markou	University of Thessaly, Greece
Jaroslav Opatrny	Concordia University, Canada
Rotem Oshman	Tel Aviv University, Israel
Marina Papatriantafilou	Chalmers University of Technology, Sweden
Joseph Peters	Simon Fraser University, Canada
Guiseppe Prencipe	Università di Pisa, Italy
Stefan Schmid	Aalborg University, Denmark and TU Berlin, Germany
Sebastien Tixeuil	LIP6, UPMC Paris 6, France
Koichi Wada	Hosei University, Japan
Yukiko Yamauchi	Kyushu University, Japan

Steering Committee

Magnus Halldorsson	Reykjavik University, Iceland
Andrzej Pelc	Université du Quebec en Outaouais, Canada
Nicola Santoro	Carleton University, Canada
Christian Scheideler	University of Paderborn, Germany
Jukka Suomela	Aalto University, Finland

Organizing Committee

Evangelos Bampas	LIF, CNRS and Aix-Marseille University, France
Jérémie Chalopin	LIF, CNRS and Aix-Marseille University, France
Shantanu Das	LIF, CNRS and Aix-Marseille University, France
Emmanuel Godard	LIF, CNRS and Aix-Marseille University, France
Damien Imbs	LIF, CNRS and Aix-Marseille University, France
Christina Karousatou	LIF, CNRS and Aix-Marseille University, France
Arnaud Labourel	LIF, CNRS and Aix-Marseille University, France

Additional Reviewers

Abeliuk, Andres
Bampas, Evangelos
Bonnet, François
Bramas, Quentin
Chalopin, Jérémie
Chitnis, Rajesh
Dereniowski, Dariusz
Di Luna, Giuseppe Antonio
Dubois, Swan
Défago, Xavier
Feuilloley, Laurent
Förster, Klaus-Tycho
Gavoille, Cyril
Hackfeld, Jan
Hopp, Alexander Vincent
Izumi, Taisuke
Karousatou, Christina
Katayama, Yoshiaki
Keramatian, Amir
Kranakis, Evangelos
Najdataei, Hannaneh
Nanongkai, Danupon
Naves, Guyslain
Nicolaou, Nicolas
Nikolakopoulos, Yiannis
Ooshita, Fukuhito
Pagli, Linda
Panagiotou, Konstantinos
Patt-Shamir, Boaz
Peleg, David
Radzik, Tomasz
Sakavalas, Dimitris
Salem, Iosif
Savic, Vladimir
Schewior, Kevin
Schlöter, Miriam
Shibata, Masahiro
Sorella, Mara
Tudor, Valentin
Uznański, Przemysław
Viglietta, Giovanni
Yu, Dongxiao

Laudatio

It is a pleasure to award the 2017 SIROCCO Prize for Innovation in Distributed Computing to Shmuel Zaks. Shmuel's contributions span an impressive range of research areas in computer science and discrete mathematics, including classic distributed computing (leader election, combinatorial and graph problems, complexity, impossibility, compact routing, self-stabilization, and more) and networking. Shmuel's work on networking has been performed during the past three decades; the first half of this period was devoted to ATM networks, and the second to optical networks.

The prize is awarded for these lifetime achievements, but especially for his pioneering research on algorithmic aspects of optical networks. In his seminal studies Shmuel formulated new problems and identified new research directions. The problems under investigation deal with a variety of aspects of optimization of the switching cost in the network, measured by the use of ADMs (ADD-DROP Multiplexers) and regenerators. Shmuel's studies deal with all algorithmic aspects of optimization problems that stem from optical networks, including the design and analysis of algorithms (e.g., approximation algorithms and on-line algorithms), complexity, parameterized complexity, and inapproximability. Shmuel's work initiated systematic studies of a variety of problems where mostly heuristics and simulations were previously used.

Examples of areas in which Shmuel's contributions to algorithmic aspects of optical networks are the most important include: ADM minimization [1, 2], regenerator placement [3, 4], traffic grooming [5], and the flex-grid model [6], where a lightpath has to be assigned a number of colors, within a contiguous or a non-contiguous range.

The 2017 Award Committee[1]

Paola Flocchini	University of Ottawa
Magnús M. Halldórsson	University of Reykjavik
Thomas Moscibroda	Microsoft Research
Andrzej Pelc (Chair)	Université du Québec en Outaouais
Christian Scheideler	University of Paderborn

[1] We wish to thank the nominators for the nomination and for contributing significantly to this text.

Selected Publications Related to Shmuel Zaks' Contribution

1. Tamar Eilam, Shlomo Moran and Shmuel Zaks, Lightpath Arrangement in Survivable Rings to Minimize the Switching Cost, IEEE Journal on Selected Areas in Communications (JSAC), special issue on WDM-based network architectures, 20(1), January 2002, pp. 172–182.
2. Tamar Eilam, Shlomo Moran and Shmuel Zaks, Approximation Algorithms for Survivable Optical Networks, Proceedings of the 14th International Workshop on Distributed Algorithms (DISC), Toledo, Spain, October 2000, pp. 104–118.
3. Michele Flammini, Alberto Marchetti Spaccamela, Gianpiero Monaco, Luca Moscardelli and Shmuel Zaks, On the Complexity of Placement of Regenerators in Optical Networks, Proc. 21st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Calgary, Canada, August 2009, pp. 154–162.
4. George B. Mertzios, Ignasi Sau, Mordechai Shalom and Shmuel Zaks, Placing Regenerators in Optical Networks to Satisfy Multiple Sets of Requests, Proc. 37th International Colloquium on Automata, Languages and Programming (ICALP), Bordeaux, France, July 2010, pp. 333–344. Best paper award.
5. Ignasi Sau, Mordechai Shalom and Shmuel Zaks, Traffic Grooming in Star Networks via Matching Techniques, Proc. 17th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Nesin Mathematics Village, Şirince, Turkey, June 2010, pp. 41–56.
6. Mordechai Shalom, Prudence W.H. Wong and Shmuel Zaks, Profit Maximization in Flex-Grid All-Optical Networks, Proc. 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Ischia, Italy, July 2013, pp. 249–260.

Invited Presentations

Online and Approximation Algorithms for Optical Networks and Scheduling

Shmuel Zaks

Department of Computer Science, Technion, Haifa, Israel
zaks@cs.technion.ac.il

Abstract. We discuss two fundamental problems that stem from optical networks models: minimizing the number of Add-Drop Multiplexers (ADMs) and regenerators. When also traffic grooming is allowed, then for a path topology network these problems can be interpreted as scheduling problems. We discuss various algorithmic aspects of several such optimization problems in offline and online settings, following [3, 4, 7, 8].

Note: This is a brief summary of the talk presented in Sirocco 2017. As such and due to space limitation, it contains neither a comprehensive reference list, nor a literature survey, nor extensions that stem from either optical networks, or scheduling theory, or the theory of algorithms; these all can be found, to a great extent, in the papers listed in the reference list.

1 Introduction – Optical Networks and Problem Definition

Background: Optical wavelength-division multiplexing (WDM) is the most promising technology today that enables us to deal with the enormous growth of traffic in communication networks, like the Internet. A communication between a pair of nodes is done via a *lightpath*, which is assigned a certain wavelength. In graph-theoretic terms, a lightpath is associated with a simple path in the network and a wavelength with a color assigned to it. We concentrate on the hardware cost, in terms of ADMs and regenerators.

ADMs: Each lightpath uses two Add-Drop Multiplexers (ADMs), one at each endpoint. If two adjacent lightpaths, i.e. lightpaths sharing a common endpoint, are assigned the same wavelength, then they can use the same ADM, provided their concatenation is a simple path. An ADM may be shared by at most two lightpaths. The total cost considered is the total number of ADMs. We thus want to color the lightpaths while minimizing the total number of ADMs, which we term the *minADM* problem. For a detailed technical explanation see [5].

Grooming: The problem of grooming is central in studies of optical networks. In graph-theoretic terms, we are given $g > 0$ and a set of simple paths, and we need to assign colors to the paths so that the union of all paths that get a particular color is a collection of disjoint simple paths in the graph, and such that at most g of them (g is

termed the grooming factor) can share the same edge. When the network topology is a path, this corresponds to scheduling problems, where the path is interpreted as the time axis, a lightpath as a job to be processed within the time interval represented by its endpoints, and the grooming factor as the number of jobs that one machine can process simultaneously. For a detailed technical explanation see [6].

Regenerators: The energy of the signal along a lightpath decreases and thus amplifiers are used every fixed distance. Yet, as the amplifiers introduce noise into the signal there is a need to place regenerators in nodes in the network along the lightpath every at most d nodes, for a given parameter d. We thus want to color the lightpaths while minimizing the total number of regenerators, which we term the *minREG* problem. As each regenerator can serve only one lightpath, the basic problem is clearly optimized by placing a regenerator after exactly d nodes on each lightpath separately. It becomes very difficult in two scenarios. The first one is when traffic grooming is allowed. The second one is when we are given p sets of lightpaths, and we want to place regenerators so as to satisfy each of these sets separately. A theoretical model was suggested in [2], where also a detailed technical explanation can be found.

Approximation Algorithms: Given an NP-hard minimization problem Π, we say that a polynomial-time algorithm $\mathcal{A}$ is an α-approximation algorithm, $\alpha \geq 1$, if for any problem instance $\mathcal{A}$ finds a feasible solution with cost at most α times the cost of an optimal solution. The class APX (Approximable) contains all NP-hard problems that can be approximated within a constant factor, and its subclass PTAS (Polynomial Time Approximation Scheme) contains the problems that can be approximated in polynomial time within a factor $1+\varepsilon$ for *any* fixed $\varepsilon > 0$; assuming $P \neq NP$ this subclass is proper. An APX-hardness result for a problem implies the non-existence of a PTAS (see [9]).

Online Algorithms: An online minimization algorithm is said to be c-competitive if for any input, it produces a solution that is at most c times that used by an optimal offline algorithm (see [1]).

2 Discussion

Four results are presented, as follows:

1. Following [8] we discuss the online version of the *minADM* problem. This corresponds to scheduling jobs to machines, where the cost is associated with opening a machine, closing a machine, or moving a machine from one job to another. We show a competitive ratio of $\frac{7}{4}$ for any network topology, including rings of size at least four, $\frac{5}{3}$ for a triangle network, and $\frac{3}{2}$ for a path topology, and show that these results are best possible.
2. Following [4] we discuss online algorithm for the *minADM* problem when grooming is allowed. The cost of a coloring is the number of ADMs; in case g lightpaths of the same wavelength enter through the same edge to one node, they can all use the same ADM (thus saving $g-1$ ADMs). This problem is NP-complete even for $g = 1$. Exact solutions are known for some specific cases, and approximation algorithms for certain topologies exist for $g = 1$. We discuss an

approximation algorithm for this problem. For every value of g the running time of the algorithm is polynomial in the input size, and its approximation ratio for a wide variety of network topologies — including the ring topology — is shown to be $2lng + o(lng)$. This is the first approximation algorithm for the grooming problem with a general grooming factor g.

3. Following [3] we discuss the *minREG* problem, when grooming is allowed. Recall that in this setting a regenerator can serve up to g lightpaths, while in scheduling this corresponds to the case in which up to g jobs can be processed simultaneously by a single machine and the goal is to assign the jobs to machines so that the total busy time is minimized. The problem is NP-hard already for $g = 2$. We discuss an algorithm whose competitive ratio is between 3 and 4.
4. Following [7], we discuss the *minREG* problem, where for given $d > 0$ and $p > 0$ sets of lightpaths, we need to place regenerators so that, for each of the p sets separately, there is a regenerator on each of its lightpaths after at most d nodes. In scheduling this corresponds to the case where we have p sets of jobs, each needs a service within d time units, and we need to place a smallest number of machines such that for each set A, each of the jobs in A is satisfied as desired. While this problem can be easily solved when $d = 1$ or $p = 1$, we discuss that for any fixed $d, p \geq 2$, it does not admit a PTAS, even if G has maximum degree at most 3 and the lightpaths have $O(d)$ lengths.

References

1. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
2. Flammini, M., Marchetti-Spaccamela, A., Monaco, G., Moscardelli, L., Zaks, S.: On the complexity of the regenerator placement problem in optical networks. IEEE/ACM Trans. Netw. **19**(2), 498–511 (2011)
3. Flammini, M., Monaco, G., Moscardelli, L., Shachnai, H., Shalom, M., Tamir, T., Zaks, S.: Minimizing total busy time in parallel scheduling with application to optical networks. Theor. Comput. Sci. **411**(40–42), 3553–3562 (2010)
4. Flammini, M., Moscardelli, L., Shalom, M., Zaks, S.: Approximating the traffic grooming problem. J. Discrete Algorithms **6**(3), 472–479 (2008)
5. Gerstel, O., Lin, P., Sasaki, G.: Wavelength assignment in a wdm ring to minimize cost of embedded sonet rings. In: INFOCOM 1998, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies (1998)
6. Gerstel, O., Ramaswami, R., Sasaki, G.: Cost effective traffic grooming in wdm rings. In: INFOCOM 1998, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies (1998)
7. Mertzios, G.B., Sau, I., Shalom, M., Zaks, S.: Placing regenerators in optical networks to satisfy multiple sets of requests. IEEE Trans. Netw. **20**(6), 1870–1879 (2012)
8. Shalom, M., Wong, P.W., Zaks, S.: Optimal on-line colorings for minimizing the number of adms in optical networks. J. of Discrete Algorithms **8**(2), 174–188 (2010)
9. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge University Press (2011)

Ignorance is Bliss (for Proving Impossibility)

Faith Ellen

Department of Computer Science, University of Toronto, Canada
faith@cs.toronto.edu

This talk surveys techniques for proving uncomputability results and lower bounds in message passing models. An uncomputability result tells us that a certain problem cannot be solved in a particular model and a lower bound tells us that a certain problem cannot be solved in a particular model when insufficient resources are available. All of these impossibility results hinge on the processes' continued ignorance of input values, network parameters, or one another's states. Ignorance can also arise from asynchrony and faults.

The first problem considered is leader election. With anonymous processes, a *symmetry argument* shows that this problem cannot be solved deterministically, even if the system is synchronous, there are no faults, message sizes are unlimited, and all processes know that the network is a cycle of a particular length [A90]. Moreover, there is no randomized algorithm for leader election if all processes know that the network is a cycle, but do not know its length [A90]. In asynchronous models where processes have distinct identifiers, if at least half the processes can crash, then leader election is shown to be uncomputable using a *partition argument* [BT85]. If fewer processes can crash, then an *adversary argument* is presented which proves that the worst case expected message complexity is quadratic in the number of processes in the network [AGV15].

A simple *reduction* from leader election shows that the consensus problem is unsolvable in an asynchronous message passing system if at least half the processes can crash. But unlike the case for leader election, consensus remains unsolvable when only one process might crash. The *valency argument* was introduced to prove this result [FLP85]. One part of the proof is a *chain argument*, which establishes the existence of a multivalent initial configuration in any consensus algorithm. The other part is an *adversary argument*, showing that from any multivalent configuration and any step which can be performed in that configuration, there is a finite execution starting from that configuration and ending with that step which results in a multivalent configuration.

For the remainder of the talk, we consider synchronous networks of nonfaulty processes with distinct identifiers. When there is no bound on the size of the messages processes can send to their neighbours, an *information theoretic argument* is used to prove that, in a cycle of even length, the number of rounds needed to colour the processes with two colours (so that no two adjacent processes have the same colour) is linear in the length of the cycle [L92]. A simple *distance argument* shows that the number of rounds necessary to determine the diameter of a network is equal to its diameter.

Finally, in the CONGEST model, where each message is restricted to contain at most B bits, a *communication complexity argument* is presented which proves that any algorithm for determining the diameter of a network with n processes requires $\Omega(n/B)$ rounds, even if all processes know that the diameter is either two or three [FHW12]. The idea is to reduce the set disjointness problem, which is known to require a linear number of bits of communication between two players, to determining the diameter of a network from a specially designed class.

Many additional examples of impossibility proofs can be found in two survey papers [L89, FR03] and a recent book [AE14].

References

[AGV15] Alistarh, D., Gelashvili, R., Vladu, A.: How to elect a leader faster than a tournament. In: Proceedings of the 34th Annual ACM Symposium on Principles of Distributed Computing, PODC 2015, pp. 365–374 (2015)

[A90] Angluin, D.: Local and global properties in networks of processors (Extended Abstract). In: Proceedings of the 12th Annual ACM Symposium on Theory of Computing, STOC 1980, pp. 82–93 (1980)

[AE14] Attiya, H., Ellen, F.: Impossibility Results for Distributed Computing. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers (2014)

[BT85] Bracha, G., Toueg, S.: Asynchronous consensus and broadcast protocols. J. ACM **32** (4), 824–840 (1985)

[FR03] Fich, F.E., Ruppert, E.: Hundreds of impossibility results for distributed computing. Distrib. Comput. **16**(2–3), 121–163 (2003)

[FLP85] Fischer, M.J., Lynch, N.A., Paterson, M.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985)

[FHW12] Frischknecht, S., Holzer, S., Wattenhofer, R.: Networks cannot compute their diameter in sublinear time. In: Proceedings of the 23rd Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2012, pp. 1150–1162 (2012)

[L92] Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**(1), 193–201 (1992)

[L89] Lynch, N.A.: A hundred impossibility proofs for distributed computing. In: Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing, PODC 1989, pp. 1–28 (1989)

Amoebots and Beyond: Models and Approaches for Programmable Matter

Christian Scheideler

Department of Computer Science, Paderborn University, Germany
scheideler@upb.de

Abstract. "Programmable matter" is a term originally coined by Toffoli and Margolus in 1991 to refer to an ensemble of fine-grained computing elements arranged in space. Since then, there has been a significant amount of work on programmable matter across multiple disciplines, including physics (e.g., crystals and complex fluids), chemistry (e.g., metamaterials and shape-changing molecules), bioengineering (DNA self-assembly and cell engineering), and robotics (modular robotics and nano robotics). So one can imagine that someday one can tailor-make programmable devices at nano-scale. However, before producing such devices, basic research is needed on the right primitives for these devices so that they can be used effectively in applications relevant for programmable matter. I will present the Amoebot model, which can be used effectively for typical applications like shape formation and coating, and discuss possible extensions of that model.

Keywords: Models · Self-organizing systems · Programmable matter

1 Introduction

Programmable matter promises to be a highly innovative technology with many interesting applications, ranging from minimal invasive surgery and adaptive materials to terraforming. Various models for programmable matter have already been proposed in different communities using approaches that differ from each other but also share some common ideas. For example, in the computational geometry community, *moteins*, whose designs are motivated by the folding behaviors of proteins, have been proposed to create complex shapes at the nanoscale [3]. Moteins consist of strings of very simple robotic modules that can fold into any volumetric shape. Additionally, motivated by computational origami, reconfigurable robots based on more complex folding primitives have been proposed. In the distributed computing community, we proposed *Amoebots* as a model for programmable matter that can adaptively form shapes and coat objects [5]. Related approaches can be found, e.g., in [8, 11]. In the DNA computing community, a number of DNA self-assembly models have been proposed. In the most basic model, the *abstract tile-assembly model* (aTAM), there are quadratic tiles with a specific glue on each side [9]. Equal glues have specific connection strengths and may bind together. Standard problems are to minimize the tile complexity (i.e., the number of different tile types) in order to form certain shapes and

to intrinsically perform computations which guide the assembly process. Whereas in the aTAM only individual tiles can be attached to an existing assembly, in more complex hierarchical assembly models, partial assemblies can also bind to each other (e.g., [4]). Beyond these passive self-assembly models, active self-assembly models based on molecular motors have been proposed, like the *nubot* model [12]. Finally, in the swarm robotics community, various prototypes of modular robots such as AMAS [10] and Mori [1] have been built in order to form complex robotic systems. In contrast to the previously mentioned models, these modular robots are computationally powerful devices. Much simpler robots have been proposed in the micro/nanorobotics field, including DNA machines, synthetic bacteria, nanoparticles, and magnetic materials, but these devices are designed and only useful for very specific tasks. More universal approaches are still under investigation.

2 The Amoebot Model

The Amoebots may form any subgraph of the infinite triangular grid $G_{\text{eqt}} = (V, E)$, where V represents all possible positions the Amoebots can occupy relative to their structure, and E represents all possible atomic movements an Amoebot can perform as well as all places where neighboring Amoebots can bond to each other. Figure 1(a) illustrates the standard planar embedding of G_{eqt}. We chose the triangular grid because in contrast to the hexagonal and square grids it guarantees that any Amoebot on the boundary of the Amoebot structure can move to an unoccupied neighboring node where it is able to bond again to a neighboring Amoebot.

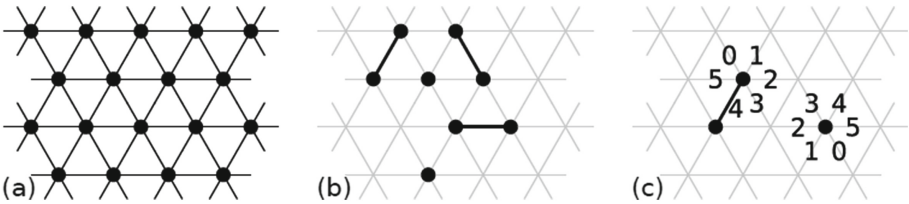

Fig. 1. (a) shows a section of G_{eqt}. Nodes of G_{eqt} are shown as black circles. (b) shows five Amoebots on G_{eqt}. The underlying graph G_{eqt} is depicted as a gray mesh. A Amoebot occupying a single node is depicted as a black circle, and a Amoebot occupying two nodes is depicted as two black circles connected by an edge. (c) depicts two Amoebots occupying two non-adjacent positions on G_{eqt}. The Amoebots have different offsets for their head port labelings.

Each Amoebot occupies either a single node or a pair of adjacent nodes in G_{eqt}, and every node can be occupied by at most one Amoebot. Two Amoebots occupying adjacent nodes are connected by a *bond*, and we refer to such Amoebots as *neighbors*. The bonds not only ensure that the Amoebot system forms a connected structure, but are also used for exchanging information.

Amoebots move through *expansions* and *contractions*: if an Amoebot occupies one node (i.e., it is *contracted*), it can expand to an unoccupied adjacent node to occupy two nodes. If an Amoebot occupies two nodes (i.e., it is *expanded*), it can contract to one of these nodes to occupy only a single node. Figure 1(b) illustrates a set of Amoebots (some contracted, some expanded) on the underlying graph G_{eqt}. We chose these kinds of movements since they allow Amoebots to stay connected while they move and it is easy to resolve movement conflicts by retracting to the contracted state. A *handover* allows two Amoebots to stay connected as they move. Two scenarios are possible: (1) a contracted Amoebot p can "push" a neighboring expanded Amoebot q and expand into a node previously occupied by q, forcing q to contract, or (2) an expanded Amoebot p can "pull" a neighboring contracted Amoebot q to a node v it occupies, causing q to expand into v and allowing p to contract.

Amoebots are *anonymous*; they have no unique identifiers. Instead, each Amoebot has a collection of *ports* — one for each edge incident to the node(s) the Amoebot occupies — that have unique labels from the Amoebot's local perspective. We assume that the Amoebots have a common *chirality* (i.e., a shared notion of *clockwise direction*), which allows each Amoebot to label its ports in clockwise order. This is justified by the assumption that Amoebots can only bond with the same face up, which is also a common assumption in DNA computing. However, Amoebots do not have a common sense of global orientation and may have different offsets for their port labels, as in Fig. 1(c). Whenever Amoebots p and q share a bond, we assume that p knows the label of q's port it bonds to and whether q's port belongs to the head or tail of q.

Each Amoebot has a constant-size local memory that can be read and written to by any neighboring Amoebot. Amoebots exchange information with their neighbors by simply writing into their memory. An Amoebot always knows whether it is contracted or expanded, and we assume that this information is also available to its neighbors (by publishing it in its local memory). Due to the constant-size memory constraint, Amoebots neither know the total number of Amoebots in the system nor any estimate of this number.

We assume the standard asynchronous model, where the Amoebot system progresses through a sequence of *Amoebot activations*; i.e., only one Amoebot is active at a time. Whenever a Amoebot is activated, it can perform an arbitrary bounded amount of computation involving its local memory and the memories of its neighbors and can perform at most one movement. A classical result under this model is that for any asynchronous concurrent execution of atomic Amoebot activations, there exists a sequential ordering of the activations which produces the same end configuration, provided conflicts which arise from the concurrent execution are resolved. We define an *asynchronous round* to be over once each Amoebot has been activated at least once. For more details, we refer to [6].

3 Future Developments

The Amoebot approach is currently explored in two further directions: One of these directions focuses on Amoebots that can sense and bond to other Amoebots but cannot

exchange information. Nevertheless, certain problems like the compression problem can be solved [2]. Another approach focuses on the problem of rearranging tiles into a specific shape using just a single Amoebot [7]. Many other variants are worth exploring since issues like fault tolerance, energy supply and consumption, and 3D structures have not been considered yet. So there is a large potential for future research in this area.

References

1. Belke, C., Paik, J.: Mori: a modular origami robot. IEEE/ASME Trans. Mechatronics (2017, to be published)
2. Cannon, S., Daymude, J., Randall, D., Richa, A.: A markov chain algorithm for compression in self-organizing particle systems. In: Proceedings of the 35th ACM Symp. on Principles of Distributed Computing, PODC 2016, pp. 279–288 (2016)
3. Cheung, K.C., Demaine, E., Bachrach, J.R., Griffith, S.: Programmable assembly with universally foldable strings (moteins). IEEE Trans. Robot. **27**(4), 718–729 (2011)
4. Demaine, E., Fekete, S., Scheffer, C., Schmidt, A.: New geometric algorithms for fully connected staged self-assembly. Theoret. Comput. Sci. **671**, 4–18 (2017)
5. Derakhshandeh, Z., Dolev, S., Gmyr, R., Richa, A., Scheideler, C., Strothmann, T.: Brief announcement: Amoebot - a new model for programmable matter. In: Proceedings of the 26th ACM Symp. on Parallelism in Algorithms and Architectures, SPAA 2014, pp. 220–222 (2014)
6. Derakhshandeh, Z., Gmyr, R., Strothmann, T., Bazzi, R., Richa, A., Scheideler, C.: Leader election and shape formation with self-organizing programmable matter. In: Proceedings of the 21st International Conference on DNA Computing and Molecular Programming, DNA 2015, pp. 117–132 (2015)
7. Gmyr, R., Kostitsyna, I., Kuhn, F., Scheideler, C., Strothmann, T.: Forming tile shapes with a single robot. In: European Workshop on Computational Geometry, EuroCG 2017 (2017)
8. Hurtado, F., Molina, E., Ramaswami, S., Sacristán, V.: Distributed reconfiguration of 2d lattice-based modular robotic systems. Auton. Robots **38**, 383–413 (2015)
9. Rothemund, P., Winfree, E.: The program-size complexity of self-assembled squares. In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, STOC 2000, pp. 459–468 (2000)
10. Terada, Y., Murata, S.: Automatic modular assembly system and its distributed control. Int. J. Robot. Res. **27**(3–4), 445–462 (2008)
11. Walter, J., Welch, J., Amato, N.: Distributed reconfiguration of metamorphic robot chains. Distrib. Comput. **17**(2), 171–189 (2004)
12. Woods, D., Chen, H.-L., Goodfriend, S., Dabby, N., Winfree, E., Yin, P.: Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In: Innovations in Theoretical Computer Science, ITCS 2016, pp. 353–354 (2013)

The Many Faces of Clock Synchronization

Christoph Lenzen

Max Planck Institute for Informatics, Saarland Informatics Campus

Abstract. Reliable and scalable clocking of hardware systems requires fault-tolerant distributed clocking methods. A brief and incomplete overview of contemporary challenges on this front is given here.

Today's hardware has a number of characteristics that renders a multi-core system or even a single chip a distributed system. A modern chip comprises billions of transistors, which work in parallel, and operates at gigahertz speeds. This entails that the system must be robust to both transient and permanent faults. Typically, the operation is synchronized, i.e., the gates' transitions are *clocked.* This requires to painfully accurately distribute a clock signal throughout the chip, as already timing deviations in the order of tens of picoseconds violate the specification under which the chip is guaranteed to operate correctly. The traditional solution are clock trees, which spread the clock signal from a single source, e.g., a quartz oscillator, throughout the chip. However, clock trees suffer from limitations in scalability and introduce a single point of failure.

Byzantine Fault-tolerant Clock Synchronization. To tackle these issues, one may employ a *clock synchronization algorithm* to synchronize multiple clock sources, which then each drive a small, local clock tree to run (possibly redundant) subsystems. Referring to these *clock domains* as nodes, a classic distributed model for clock synchronization arises. We assume (i) a fully connected system of n nodes, (ii) at most $f < n/3$ *Byzantine faults* (i.e., arbitrary behavior), (iii) for each node a local clock source of *bounded drift* (i.e., at all times running at a rate between 1 and $\vartheta > 1$), and (iv) *bounded delay,* i.e., each message takes between $d - \varepsilon$ and d time to arrive (where computational delay is accounted for in d).

A classic algorithm in this model is due to Srikanth and Toueg [12], achieving a *skew* of $2d$, i.e., the maximum time difference between corresponding clock "ticks" of non-faulty nodes is $d + \varepsilon$. It has been successfully implemented in hardware [5]. This solution has two crucial downsides. First, the skew lower bound of $\Omega(\varepsilon)$ [10] is not matched, and second the algorithm is not *self-stabilizing,* i.e., correct operation is not re-established after (an unbounded number of) transient faults.

Pulse Generation. With self-stabilization as an additional requirement, the problem has been studied under the name Byzantine fault-tolerant self-stabilizing *pulse generation.* Due to the substantial challenge this imposes, many algorithms need to communicate large amounts of information. Thus, the amounts of bandwidth per node (number of broadcasted bits per time unit) becomes an additional quality measure. The currently best algorithms result from a recent framework for reducing the problem to

synchronous consensus was presented [9], resulting in a solution with stabilization time and bandwidth polylog$f$. This reduction "translates" the running time of the consensus algorithm to stabilization time and its message size to bandwidth, with at most a factor $O(\log f)$ increase in each.

Open Problem 1 *Is pulse synchronization at least as hard as consensus?*

Metastability. The second issue of the Srikanth-Toueg algorithm, the potentially large gap between the achieved skew and the lower bound of $\Omega(\varepsilon)$, has been addressed by another classic synchronization algorithm, due to Lundelius Welch and Lynch [13]. We implemented a variant of this algorithm in hardware [7], with surprisingly good results. However, the achieved precision is still by roughly an order of magnitude worse than that of a clock tree, necessitating further improvements. Doing so requires to overcome a fundamental obstacle: *metastability.* Metastability is an unstable equilibrium state of a bistable element (such as a register or flip-flop) that may occur when input signals are "unclean" and violate timing constraints. Marino proved [11] that the possibility of metastability cannot be avoided when measuring the skew between different clocks, which naturally is necessary for any clock synchronization algorithm.

Usually, metastability is handled by simply waiting for the register, flip-flop, etc. to recover a stable state with sufficiently high probability. However, when trying to run the Lynch-Welch algorithm at sufficiently high frequency to achieve the desired skew bound, there is insufficient time for this solution. Another issue is that a faulty node could provide out-of-spec input signals to a correct node, possibly "infecting" it with metastability.

We introduced an approach based on Kleene logic [4] modelling metastability in a worst-case fashion. It represents "unclean" signals (intermediate voltages between logical 0 and 1, oscillations, late transitions, etc.) by a third symbol M and extends the gate function $f : \{0,1\}^n \to \{0,1\}$ of a basic gate to its *metastable closure* $f_M : \{0,1,M\}^n \to \{0,1,M\}$. Defining for $x \in \{0,1,M\}^n$ that $Res(x) := \{x' \in \{0,1\}^n \,|\, x'_i \neq x_i \Rightarrow x_i \neq M\}$, i.e., the set of strings $x \in \{0,1\}^n$ obtained by treating M as "wildcard," f_M is given by $f_M(x) = b \in \{0,1\}$ iff $f(x') = b$ for all $x' \in Res(x)$ and $f_M(x) = M$ else. It was shown that for any Boolean function f, f_M can be implemented using standard logic [4]. In particular, this makes it possible to implement the Lynch-Welch algorithm in a way that avoids metastability-induced faults deterministically. However, in general a circuit implementing f_M may be of exponential size in n, even if one for f is small.

Open Problem 2 *For a Boolean function f, how large is a circuit that implements f_M on all inputs with up to k metastable bits?*

Combining Self-stabilization and High Precision. In [6], it is shown how to couple (a variant of) the Lynch-Welch algorithm with the solution from [2] to obtain a clock synchronization algorithm of skew $O(\varepsilon)$ and stabilization time $O(f)$. Unfortunately, this results in a significant increase in the circuitry for a single node, which is likely to increase the probability that an individual node fails. Therefore, it is desirable to find a

different coupling mechanism that ensures that the underlying pulse synchronization algorithm is only relied on during stabilization.

Open Problem 3 *Find coupling mechanisms that enable self-stabilization of the Lynch-Welch algorithm without interfering with it after stabilization.*

Distributing the Clock Signal. All of the above solutions assume full connectivity, which is impractical across a whole system. Thus, a way of distributing the clock signal reliably using a low-degree topology is required. HEX [1, 8] provides a first stab at the problem, by ensuring both self-stabilization and resilience to one Byzantine fault in the neighborhood of each node. More generally, one may aim at tolerating $\Omega(\Delta)$ such *local* faults in degree-Δ networks. However, HEX is not sufficiently precise to be considered as general purpose clocking method.

Open Problem 4 *Come up with competitive low-degree clock distribution networks that combine self-stabilization and tolerate local Byzantine faults.*

For further details on these challenges and others, please refer to our survey [3].

References

1. Dolev, D., Fuegger, M., Lenzen, C., Perner, M., Schmid, U.: HEX: scaling honeycombs is easier than scaling clock trees. In: Proceedings Symposium on Parallelism in Algorithms and Architectures, SPAA 2013 (2013)
2. Dolev, D., Fuegger, M., Lenzen, C., Schmid, U.: Fault-tolerant algorithms for tick-generation in asynchronous logic: robust pulse generation. J. ACM **61**(5), 30:1–30:74 (2014)
3. Dolev, D., Fugger, M., Lenzen, C., Schmid, U., Steininger, A.: Fault-tolerant distributed systems in hardware. Bull. EATCS **116** (2015)
4. Friedrichs, S., Függer, M., Lenzen, C.: Metastability-containing circuits (2016). CoRR abs/1606.06570
5. Függer, M., Schmid, U.: Reconciling fault-tolerant distributed computing and systems-on-chip. Distrib. Comput. **24**(6), 323–355 (2012)
6. Khanchandani, P., Lenzen, C.: Self-stabilizing Byzantine clock synchronization with optimal precision. In: Proceedings of Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2016 (2016)
7. Kinali, A., Huemer, F., Lenzen, C.: Faul-tolerant clock synchronization with high precision. In: Symposium on VLSI, ISVLSI 2016 (2016)
8. Lenzen, C., Perner, M., Sigl, M., Schmid, U.: Byzantine self-stabilizing clock distribution with HEX: implementation, simulation, clock multiplication. In: Proceedings of Conference on Dependability, DEPEND 2013 (2013)
9. Lenzen, C., Rybicki, J.: Self-stabilising byzantine clock synchronisation is almost as easy as consensus. In: Proceedings of 31st Symposium on Distributed Computing, DISC 2017 (2017, to appear)
10. Lundelius, J., Lynch, N.: An upper and lower bound for clock synchronization. Inf. Comput. **62**(2–3), 190–204 (1984)

11. Marino, L.: General theory of metastable operation. IEEE Transac. Comput. **C-30**(2), 107–115 (1981)
12. Srikanth, T.K., Toueg, S.: Optimal clock synchronization. J. ACM **34**(3), 626–645 (1987)
13. Welch, J.L., Lynch, N.A.: A new fault-tolerant algorithm for clock synchronization. Inf. Comput. **77**(1), 1–36 (1988)

Contents

Probabilistic Algorithms

Computational Complexity

Dynamic Networks