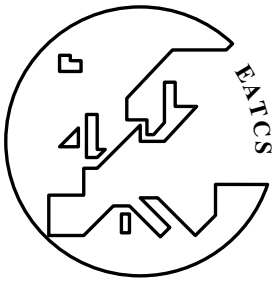
ISSN 0252–9742

Bulletin

of the

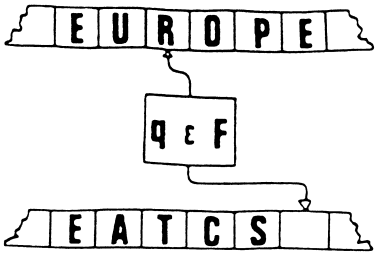
European Association for Theoretical Computer Science

EATCS



Number 115 February 2015

Council of the European Association for Theoretical Computer Science



President:	Luca Aceto	Iceland
Vice Presidents:	Paul Spirakis	Greece
	Burkhard Monien	Germany
	Antonin Kucera	Czech Republic
Treasurer:	Dirk Janssens	Belgium
Bulletin Editor:	Kazuo Iwama	Kyoto, Japan

Lars Arge	Denmark	Elvira Mayordomo	Spain
Jos Baeten	The Netherlands	Luke Ong	UK
Paul Beame	USA	Catuscia Palamidessi	France
Mikolaj Bojanczyk	Poland	David Peleg	Israel
Josep Díaz	Spain	Giuseppe Persiano	Italy
Zoltán Ésik	Hungary	Alberto Policriti	Italy
Fedor Fomin	Norway	Alberto Marchetti Spaccamela	Italy
Pierre Fraigniaud	France	Vladimiro Sassone	UK
Leslie Ann Goldberg	UK	Roger Wattenhofer	Switzerland
Monika Henzinger	Austria	Thomas Wilke	Germany
Christos Kaklamanis	Greece	Peter Widmayer	Switzerland
Juhani Karhumaki	Finland	Gerhard Wöeginger	The Netherlands

Past Presidents:

Maurice Nivat	(1972–1977)	Mike Paterson	(1977–1979)
Arto Salomaa	(1979–1985)	Grzegorz Rozenberg	(1985–1994)
Wilfred Brauer	(1994–1997)	Josep Díaz	(1997–2002)
Mogens Nielsen	(2002–2006)	Giorgio Ausiello	(2006–2009)
Burkhard Monien	(2009–2012)		

Secretary Office:	Ioannis Chatzigiannakis	Greece
	Efi Chita	Greece

EATCS Council Members

email addresses

Luca Aceto . . . . . luca@ru.is
Lars Arge . . . . . large@madalgo.au.dk
Jos Baeten . . . . . jos.baeten@cwi.nl
Paul Beame . . . . . beame@cs.washington.edu
Mikolaj Bojanczyk . . . . . bojan@mimuw.edu.pl
Josep Díaz . . . . . diaz@lsi.upc.es
Zoltán Ésik . . . . . ze@inf.u-szeged.hu
Fedor Fomin . . . . . fomin@ii.uib.no
Pierre Fraigniaud . . pierre.fraigniaud@liafa.univ-paris-diderot.fr
Leslie Ann Goldberg . . . . . leslie.goldberg@cs.ox.ac.uk
Monika Henzinger . . . . . monika.henzinger@univie.ac.at
Dirk Janssens . . . . . Dirk.Janssens@ua.ac.be
Christos Kaklamanis . . . . . kakl@ceid.upatras.gr
Juhani Karhumäki . . . . . karhumak@cs.utu.fi
Antonin Kucera . . . . . tony@fi.muni.cz
Elvira Mayordomo . . . . . elvira@unizar.es
Burkhard Monien . . . . . bm@uni-paderborn.de
Luke Ong . . . . . luke.Ong@cs.ox.a.uk
Catuscia Palamidessi . . . . . catuscia@lix.polytechnique.fr
David Peleg . . . . . peleg@wisdom.weizmann.ac.il
Giuseppe Persiano . . . . . giuper@dia.unisa.it
Alberto Policriti . . . . . alberto.policriti@uniud.it
Alberto Marchetti Spaccamela . . . . . alberto@dis.uniroma1.it
Vladimiro Sassone . . . . . vs@ecs.soton.ac.uk
Roger Wattenhofer . . . . . wattenhofer@tik.ee.ethz.ch
Thomas Wilke . . . . . thomas.wilke@email.uni-kiel.de
Peter Widmayer . . . . . widmayer@inf.ethz.ch
Gerhard Wöeginger . . . . . g.j.woeginger@math.utwente.nl

Bulletin Editor: Kazuo Iwama, Kyoto, Japan
Cartoons: DADARA, Amsterdam, The Netherlands

The bulletin is entirely typeset by pdfTEX and ConTEXt in TXfonts.

All contributions are to be sent electronically to

bulletin@eatcs.org

and must be prepared in LaTeX 2ε using the class `beatcs.cls` (a version of the standard LaTeX 2ε `article` class). All sources, including figures, and a reference PDF version must be bundled in a ZIP file.

Pictures are accepted in EPS, JPG, PNG, TIFF, MOV or, preferably, in PDF. Photographic reports from conferences must be arranged in ZIP files layed out according to the format described at the Bulletin's web site. Please, consult `http://www.eatcs.org/bulletin/howToSubmit.html`.

We regret we are unfortunately not able to accept submissions in other formats, or indeed submission not *strictly* adhering to the page and font layout set out in `beatcs.cls`. We shall also not be able to include contributions not typeset at camera-ready quality.

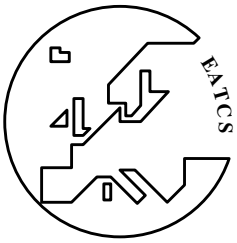
The details can be found at `http://www.eatcs.org/bulletin`, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email `bulletin@eatcs.org`.

Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

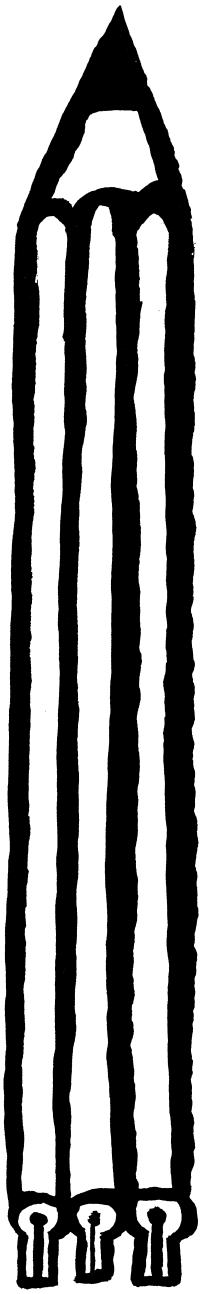
The Editor welcomes proposals for surveys, tutorials, and thematic issues of the Bulletin dedicated to currently hot topics, as well as suggestions for new regular sections.

Table of Contents

EATCS Matters



Letter from the President

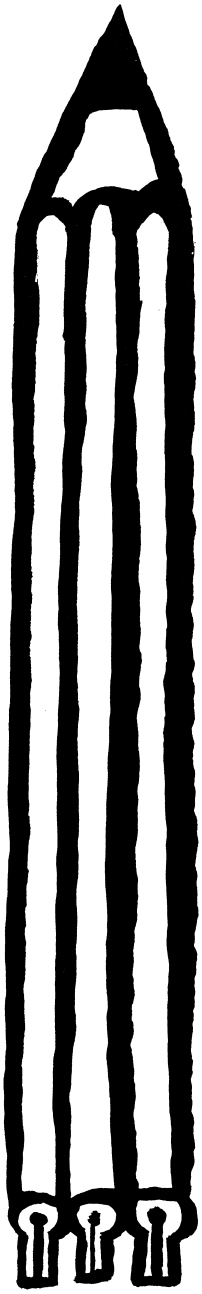


Dear colleagues,

First of all, let me wish you a belated happy 2015. I hope that this will be a healthy and fruitful year for all of you, and I look forward to working together with all of you in order to continue promoting the development of theoretical computer science, broadly construed, and to increase the influence of the EATCS within the theoretical-computer-science community.

I hope that many of you will submit their best work to ICALP 2015 (deadline: Tuesday, 17 February 2015). As you know, ICALP 2015 will be held in Kyoto, Japan, and will be co-located with LICS 2015. It promises to be a mouthwatering scientific meeting, with five invited talks (two joint with LICS 2015) delivered by Ken Kawarabayashi (NII, Japan), Valerie King (University of Victoria, Canada), Thomas Moscibroda (MSR Asia, China), Anca Muscholl (Université Bordeaux, France) and Peter O'Hearn (Facebook, UK) as well as a masterclass on algoritms and complexity for Japanese puzzles by Ryuhei Uehara (JAIST, Japan) and three invited tutorials (joint with LICS) that will be presented by Piotr Indyk (MIT, USA), Andrew Pitts (University of Cambridge, UK) and Geoffrey Smith (Florida International University, USA).

I am happy to inform you that the EATCS Awards Committee 2015 consisting of Fedor Fomin, Kim G. Larsen and Vladimiro Sassone (chair) has selected Christos Papadimitriou (UC Berkeley, USA) as the recipient of the EATCS Award 2015. Congratulations to Christos! The award, which is given to acknowledge extensive and widely recognized

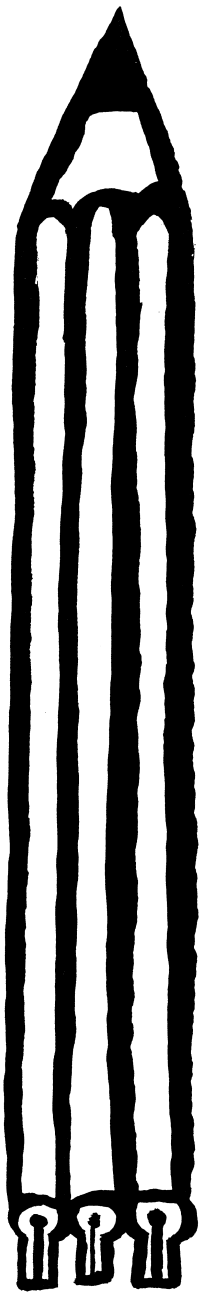


contributions to theoretical computer science over a life-long scientific career, will be presented to Christos at ICALP 2015. Moreover, during the conference, we will hand out the Presburger Award 2015 and honour the EATCS Fellows vintage 2015.

If that weren't enough to whet your scientific appetite, ICALP 2015 will also feature a couple of events to celebrate the 40th anniversary of the journal Theoretical Computer Science. More details about this celebration will be available in due course on the ICALP/LICS 2015 web site at http://www.kurims.kyoto-u.ac.jp/icalp-lics2015/.

ICALP 2016 will be held in Rome, Italy, and Tiziana Calamoneri and her team are already busy working on the organization of the conference. The PC chairs for ICALP 2016 will be Yuval Rabani (Track A), Davide Sangiorgi (Track B) and Michael Mitzenmacher (Track C). On behalf of the community, I thank these colleagues for accepting this demanding responsibility. As usual, the call for papers for ICALP 2016 will be ready for distribution at ICALP 2015.

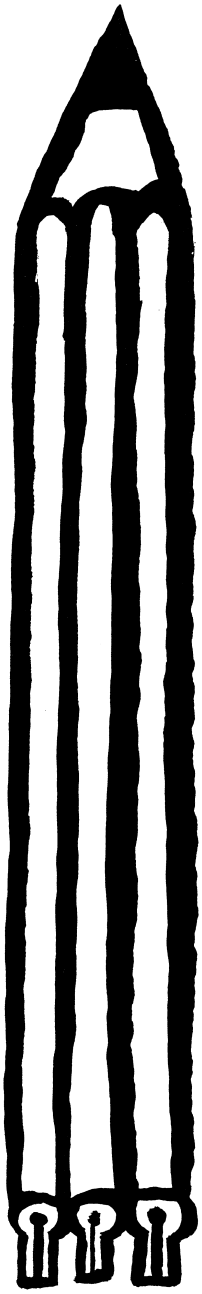
As you might know already, from ICALP 2016 there will be an important change in the publication of the proceedings of ICALP. Since the contract between the EATCS and Springer will expire at the end of 2015, the council of the EATCS has discussed the publication outlet for the proceedings of ICALP from 2016. Based on the discussion and on the ensuing vote, the council of the EATCS has decided that, starting from ICALP 2016, the proceedings of ICALP will be published in open-access form in LIPIcs (Leibniz International Proceedings in Informatics).



The EATCS has had a long and good working relationship with Springer, which we really appreciate. However, the council overwhelmingly thought that it is time for a change, as far as the publication of the proceedings of ICALP is concerned, and that it was best to make this decision well before the end of our current contract, which we will, of course, honour. We hope that this new development will be good for the EATCS, and that it will be popular with our members and with the TCS community at large.

As announced in the piece authored by Edward Hirsch that appears in this issue of the Bulletin, our association now has a Russian Chapter. This is the first chapter of the EATCS in Eastern Europe and I hope that it will help the development of TCS research and education in Russia. Having a chapter of the EATCS might also further strengthen the Russian community of TCS researchers and its ties with researchers elsewhere. I am truly excited by this development and I wish Edward and the other funding members of the Russian Chapter the best of luck with their work in making their chapter an important voice in Russian computer science.

The council of the EATCS continues to take steps in order to promote theoretical computer science at large and to support young researchers. This year we will present the first EATCS Distinguished Dissertation Awards to outstanding theses in theoretical computer science. The award committee, consisting of Javier Esparza, Fedor Fomin, Luke Ong and Giuseppe Persiano (chair), is busy examining the excellent nominations they have received. The EATCS



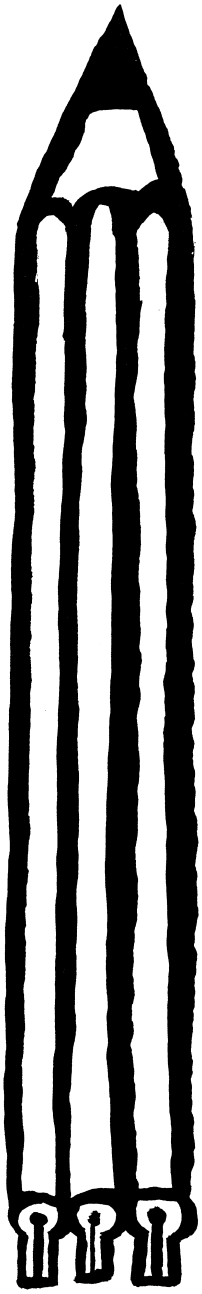
looks forward to honouring the first recipients of this new award for young scientists.

The second EATCS Young Researcher School will be the TOPDRIM International School on Topology and Data: Understanding Complexity and Concurrency through Topology, *which will be organized by Emanuela Merelli in Camerino, Italy, in the period July 13-22, 2015.*

The EATCS Young Researchers School Liaison Committee, chaired by Antonin Kucera, welcomes applications from organizers of graduate schools in theoretical computer science. If you are organizing a young-researcher school next year, please drop Antonin a line.

I hope that you will appreciate the steps that the council of the EATCS has taken on several fronts, even though there is still much more that we could be do if we had suitable resources. I am truly grateful to our institutional sponsors and to our members for their support over the years. It is only with your support that we can continue producing our Bulletin (which is freely accessible to everyone), and that we can support awards, young research schools and other worthwhile activities in TCS. Our goal is to give back to the TCS community what the community gives us by supporting the EATCS with their membership fees. If you are not already a member, I hope that you will join the EATCS and encourage your colleagues and students to do so.

Of course, the EATCS is only one of a collection of sister associations whose goal is to foster the development of TCS research. In order to encourage double



registration, we are offering a discount to members of sister associations with which we have a reciprocity agreement: EACSL, ACM SIGACT and ACM SIGLOG members can join the EATCS for 25 Euros per year (two years for PhD students). In return, those associations offer a discount to members of the EATCS.

As usual, let me close this letter by reminding you that you are always most welcome to send me your comments, criticisms and suggestions for improving the impact of the EATCS on the theoretical-computer-science community at president@eatcs.org. We are here to serve the theoretical computer science community and we will consider all your suggestions and criticisms carefully.

I look forward to hearing from you.

Luca Aceto, Reykjavik, Iceland
February 2015

Letter from the Bulletin Editor

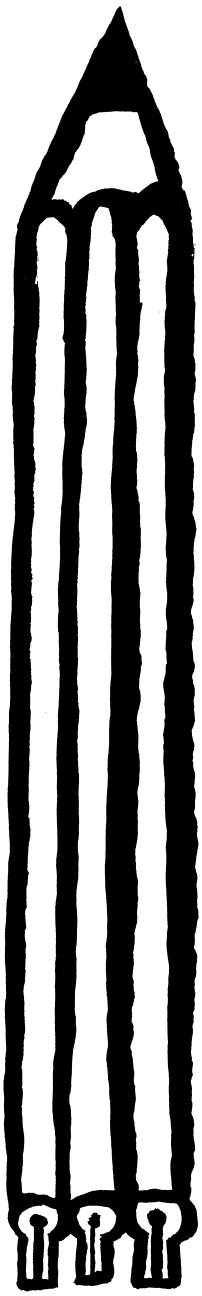
Dear Reader,

Last December I was in Zurich for a week to attend a PhD defense at ETH. Whenever I came to Switzerland, I was always reminded that it is an expensive country. Then in January, we had the surprising news that Swiss government stopped its currency control and Swiss Franc immediately hiked by some 20%. Even more expensive by another 20%. Huuu...

I was working on online currency exchange problems more than a decade ago. The basic algorithm for one-way trading is quite simple: Just setting a min and a max prices (meaning the price never gets more than the max and can fall to the min unexpectedly at any time), then we can automatically obtain an optimal algorithm that determines the amount of money to be exchanged each day. "One way" means that the term is an increasing or a decreasing term. The former means the price (of e.g., Swiss Franc) is roughly increasing during the term. Hence if we can guess only three things, max, min and a direction of the price correctly, then we are done.

It is often said that theory does not help in practice. However, due to our 2,3-year old experiment, this algorithm is surprisingly good and I just remembered that when I saw the above news. AND that night, I had a dream in which I became a millionaire instantly by currency trading. In Japan there is a saying that your first dream of the year often comes true. Mine is not exactly the first one, but almost...

This February issue also has a big news. We welcome two new column editors, Stefan



Schmid for the DC column and Juraj Hromkovic for Education column. Panagiota Fatourou who has given us fantastic surveys for more than six years will be resigning after the next issue. So Feb and June issues include a joint DC column; you can see their short statements after the cover page of this issue. I do not know well, but the Education column is not completely new, either, but it has been quiet for a long time. Everyone knows that Juraj is a real expert in this field; he sent me his four textbooks for junior high and high school students in Switzerland. I do not read German, but I immediately thought that they will be translated to many other languages, hopefully to Japanese, too.

I have another news that is a bit sad. Cris Calude, who has been sending us his "News from New Zealand" for 21 years, says his column in this issue will be the last one. I know many people were looking forward to reading his conversation series, including myself. It is amazing to me that the whole sentences do not sound like those in scientific journals, but rather like those in general magazines targeting upper-level intellectuals. Thank you very much, Cris.

I have another announcement. Luca suggested a special section for the centenary of the birth of George Dantzig and Richard Hamming. This is a fantastic idea and I wanted to start it from this issue. Unfortunately we could not make it, but I am sure it will start from the next issue. All types of essays, both technical and non-technical, are welcome; please send them to me as soon as possible.

Kazuo Iwama, Kyoto
February 2015

The EATCS Award 2015 is awarded to

Christos Papadimitriou

for his visionary and pioneering contributions to the development of Computer Science and its connections to the physical and applied sciences.

Prof Christos H. Papadimitriou (henceforth Christos) had the most fundamental effect on the shape of several, major research areas within Theoretical Computer Science. He is an intellectual giant interested in the fundamental problems of many scientific disciplines and their computational origins, he was able throughout his exceptional career to draw connections between areas at the highest and deepest intellectual level. His work is characterised by opening new research directions, defining new models, asking fundamental questions, and developing sophisticated techniques to deal with such problems.

Christos had been a central contributor to the fields of *algorithms and complexity*. His most influential work therein concerns the complexity of approximation problems. The question of understanding which decision problems in NP can be computed in polynomial time and which are NP-complete had become an elegant theory by the late nineteen eighties. But after much effort the analogous question of having a similar theory for approximation problems was still wide open. At this time Christos (with Yannakakis) defined the class *MaxSNP*, which turned out to be key to the discovery of probabilistically checkable proofs and, finally, helped place inapproximability problems within the classical NP theory. A second line of his pivotal contributions is establishing the complexity theory of local search. It appeared that complexity classes of total functions PLS, PAD, PPAD defined by Christos are key to understanding the computational power of fundamental mathematical principles and are deeply related to proof complexity.

Christos has been a towering influence on *algorithmic economics*, the important and growing field at the interface between Economics and Computer Science. His influence on the economics of the Internet, and on the way classical concepts in economics should be viewed when agents are computationally bounded, cannot be overestimated. Among his many results his work (with Koutsoupias) on the "*price of anarchy*" has been enormously influential. Perhaps his most striking single contribution in the area was establishing (with Daskalakis and Goldberg) the completeness of

the Nash equilibrium problem in the class PPAD.

Christos contributed numerous important results to *Operations Research*. They include general efficient treatment of multi-objective criteria, new approximation algorithms to NP-complete problems such as the Traveling Salesman Problem, and an ingenious way to solve linear programs with exponentially many constraints.

Although we cannot hope to make justice to the whole of his research here, let us mention his many contributions to *Database theory*, in particular on efficient implementations of different queries to databases, and to *Online algorithms*, where he solved the hottest, long standing problem of this field at the time: the k-server conjecture.

As an ambassador of computer science to other fields, Christos took on the queen of problems of *Evolutionary Biology*, namely, the problem of sex. To date, no satisfactory explanation exists for the nearly universal prevalence of sexual reproduction, whereas it seems that asexual reproduction better supports Darwin's theory of maximising fitness. In highly original work, Christos and his collaborators suggest a new "mixability theory" which argues that sexual reproduction drives a different survival advantage, viz., the ability of alleles to thrive in a variety of genetic contexts.

Christos's work has been of consistently high quality for over 30 years. His books and papers transformed the way people thought about their research, providing fresh insight, research direction or powerful technique. He has provided the field with some excellent textbooks, e.g., on Algorithms, Complexity, Optimisation and Database theory, and has a very strong record of mentorship to young scientists. His expository work through his novel "Turing" and the fascinating "Logicomix" completes his profile as a true "*Rennaissance man"* of Computer Science.

For all these reasons, the EATCS wants to celebrate Prof. Christos H. Papadimitriou, his technical brilliance, inspirational figure, wide reaching and profoundly impactful work, and is honoured to award him with its most prestigious prize.

Fedor V. Fomin, Kim Larsen, Vladimiro Sassone

Institutional Sponsors

CTI, Computer Technology Institute & Press "Diophantus"
Patras, Greece

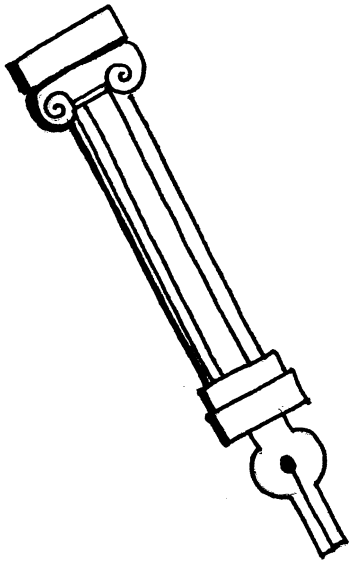
CWI, Centum Wiskunde & Informatica
Amsterdam, The Netherlands

MADALGO, Center for Massive Data Algorithmics
Aarhus, Denmark

Microsoft Research Cambridge
Cambridge, United Kingdom

Springer-Verlag
Heidelberg, Germany

EATCS
Columns



THE ALGORITHMICS COLUMN

BY

GERHARD J WOEGINGER

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
gwoegi@win.tue.nl

BEATCS no 115

K-BEST ENUMERATION

David Eppstein*

Abstract

We survey k-best enumeration problems and the algorithms for solving them, including in particular the problems of finding the k shortest paths, k smallest spanning trees, and k best matchings in weighted graphs.

1 Introduction

Researchers in the design and analysis of algorithms have had much success in finding precise mathematical formulations of optimization problems, developing efficient algorithms for solving those problems, and proving worst-case bounds on the time complexity of these algorithms. A case in point is the problem of finding minimum spanning trees in graphs, studied since the 1920s [22] and finally shown in 1995 to be solvable in (randomized) linear time [114]. Other problems of a similar nature are the graph shortest path problem, famously solved by Dijkstra's algorithm [57], and the problem of finding minimum weight matchings in graphs, first solved in polynomial time by Edmonds [62] and later significantly improved [82, 83].

However, many real-world problems are only approximately modeled by these mathematical formulations. The most desirable solution may depend on quality criteria that are more complicated than a simple sum of edge weights, it may be determined by data that is not yet available or subject to unpredictable changes, or its choice may involve political or social processes that are not well-modeled by mathematical values. In such a situation, it is desirable for an algorithm to output more than one candidate solution. Once such a list of candidates is generated, one can apply more sophisticated quality criteria, wait for data to become available to choose among them, or present them all to human decision-makers. On the other hand, the exponential growth of the solution space for many combinatorial

*Computer Science Department, University of California, Irvine; Irvine, CA, USA. This material is based upon work supported by the National Science Foundation under Grant CCF-1228639 and by the Office of Naval Research under Grant No. N00014-08-1-1015. A significantly shorter version of this material appears in the Springer *Encyclopedia of Algorithms*, 2014.

optimization problems would make it infeasible to list all possible candidate solutions; instead, some filtering is needed.

A common approach to such problems breaks the decision process into two stages. In the first stage, the problem is modeled mathematically in the standard way (e.g. for the problems above, by weighting the edges of a graph and seeking a solution with a small sum of edge weights). However, rather than finding a single optimal solution, an algorithm is used to find the *k best solutions*, for a given parameter value k: that is, the algorithm finds a set of k different solutions that have a better solution quality than any solution not in the set. Then, these candidates are passed on to the second stage of the decision process, where they may be evaluated and compared using more complicated evaluation criteria than sums of weights, used as a set of candidates to switch among dynamically based on updated data, or passed to human decision-makers. A familiar example of this occurs in car navigation systems, which can typically be programmed to present drivers with multiple alternative routes between the current location of the car and the target destination.

In this review we survey algorithms for generating sets of the k best solutions, for several optimization problems, as well as the applications of these algorithms.

2 General Principles

If the single best solution to an optimization problem can be found by an algorithm whose running time is polynomial in the input size, then it is typical for the k best solutions to be found in time polynomial in both the input size and k. There are two general techniques that can be used to achieve this, based on optimal substructures and solution space partitioning. In turn, these methods rely on fast algorithms for the *selection problem* in certain sets of structured values.

2.1 Selection

Underlying many k-best algorithms is the problem of *selection*, finding the k smallest values from a larger set of values [21]. If there are n values in the set, then selection may be solved in time $O(n + k)$, for instance by *quickselect*, an algorithm that chooses a random pivot value from the set, partitions the rest of the set into subsets that are less than, equal to, or greater than the pivot, and then recurses into one of these subsets [131].

However, in k-best problems, we do not wish to generate all solutions before finding the best of them: we wish to avoid the $O(n)$ part of this $O(n+k)$ time bound. Often, the solution values among which we are selecting the k best have some additional structure that allows finding the k best without examining all solutions.

For instance, suppose that the space of solutions can be represented as the vertex set of an (implicitly defined) edge-weighted graph, that the degrees of the vertices in this graph are bounded, and that the quality of any particular solution equals the length of the shortest path in this graph from some designated starting vertex to the solution vertex. In such a case, Dijkstra's algorithm can be used to recover the k smallest of these values in time $O(k \log k)$, independent of the size of the graph, without requiring that the whole graph be explored. Other structures that allow the k best solutions to be found more efficiently than enumeration of the whole solution set include representations of the solution space as the set of sums of pairs of values drawn from two sorted sets [76, 111], or as the elements of an (implicitly defined) matrix whose rows and columns are sorted [77]. Again, for such structures, the k smallest values can be found efficiently without examining the whole matrix or the whole set of pair sums.

A general framework that can be used to describe many of these structured selection problems is the problem of selection in a d-ary heap. A min-heap is a rooted tree, with values associated with its nodes, such that each non-root node has a value that is at least as large as its parent. In order to avoid having to construct all solutions before performing a selection algorithm, and in order to handle situations in which this tree has infinitely many nodes, we will assume that our selection algorithm is given as input an implicit representation of such a heap. That is, its input consists of a binary representation of the root node of the heap together with pointers to two subroutines that take a single node as input: one subroutine to list the children of any node in the heap, and another subroutine to find the value of any node in the heap. Based on this information, the problem is to find the k nodes with the smallest values. For instance, the problem of selection in a sorted array can be represented in this way by a heap in which each node represents an array cell, the root is the upper left corner of the array, and the parent of each node is either the cell above it (if it is not in the first row of the array) or to its left (if it is in the first row). An implicit min-heap represented in this way is a special case of a graph (where the weight of an edge is the difference in values between a node and its parent), so Dijkstra's algorithm can find the k best solution values in such a structure in time $O(k \log k)$. A more sophisticated algorithm of Frederickson for searching an implicit heap improves this time bound to $O(k)$ [74].

2.2 Optimal Substructures

The optimal substructure technique is most applicable to problems whose optimization version is solved by a dynamic programming algorithm. In the dynamic programming technique, one identifies a family of polynomially many subproblems of the same type as the original problem, and a recurrence relation by which the value of any subproblem can be calculated in terms of the values of smaller sub-

problems. The value of the whole problem can then be found by iterating through all of the subproblems in order by their sizes, using the recurrence to calculate and store the value of each subproblem as a combination of previously-computed values. To apply this algorithmic framework to a k-best problem, one stores the k best solutions to each subproblem (rather than a single best solution), and modifies the recurrence to compute these k best solutions as combinations of the k best solutions of smaller subproblems. The k-best algorithm then iterates through the subproblems in order by size, using this recurrence to compute the k best values of each subproblem.

As an example, the problem of finding a minimum weight triangulation of a point set (NP-hard for arbitrary planar point sets [141]) can be solved in polynomial time for points in convex position, by a dynamic program in which the subproblems are the subsets of the points that can be formed by intersecting them with a half-plane [84,118]. The optimal solution for such a problem may be found by choosing one edge of the convex hull, testing all triangles that could be based on that edge, and for each such triangle adding the weights of the optimal solutions of the two subproblems that the triangle splits the problem into. For n input points, there are $O(n^2)$ subproblems, the solution to each of which involves examining $O(n)$ triangles, so the total time is $O(n^3)$. To modify this dynamic programming algorithm to find the k smallest-weight triangulations, we may store the k smallest weight solutions for each subproblem (in sorted order). To compute the value of a problem, we again choose one convex hull edge and test all triangles that include that edge. However, a single triangle may produce $O(k^2)$ solutions (combinations of the k values stored in each of the two subproblems on either side of the triangle), so implemented naively this method would take $O(k^2n)$ time per subproblem to sort through all these solutions and select the k best of them, giving an $O(k^2n^3)$ overall time bound. To speed this up, one can replace the computation for each triangle by an algorithm for finding the k smallest values among the pairwise sums from two sets of k sorted values, which may be solved in $O(k)$ time [76]. This speedup leads to an $O(kn^3)$ time bound overall.

2.3 Solution-Space Partitions

An alternative general approach to k-best problems involves partitioning the space of solutions into smaller subspaces defined by subproblems of the original problem. For instance, in a problem where the solutions may be described as sets of edges in a graph, a subproblem may be specified by requiring some edges to be included in any solution while removing some other edges from the graph, forcing them to be excluded.

If it is possible to find not just the best solution to a given optimization problem, but also the second best, then this may be used to partition the space of solutions

into a collection of subproblems that have the structure of a binary heap. Each node of this heap represents a single subproblem (represented e.g. by its sets of forbidden and required edges) and has a value equal to the second-best solution within that subspace. The root of the heap is the space of all solutions, i.e., a subproblem where we have not yet made any restrictions. For a given node X of this heap, the best and second-best solution must differ from each other, for instance by including an edge e in one of these two solutions and excluding e from the other solution. The two children of node X may then be formed as the subproblems where e is added respectively to the set of forbidden or required edges. One of these two subproblems includes the best but not the second best solution, and the other includes the second best but not the best solution, so both subproblems have second-best solution values that differ from the value for the whole space. This hierarchical partition of the solution space organizes all the solutions except the best one into an implicit binary heap, allowing the selection algorithm of Frederickson [74] to be used to find the k best solutions while examining only $O(k)$ of the subproblems from this structure. This, the time for this procedure is $O(k)$ times the time to find a single second-best solution.

An alternative partitioning technique uses only the best solution in each subproblem, rather than also requiring the second best solution. However, in exchange for this easier computation in each subproblem, it produces a heap-ordered tree of solution values in which the degree of each tree node is higher. Suppose, for instance, that the k-best problem that we are trying to solve has an input in the form of a graph, and that its solutions can be represented by sequences of edges in this graph; for instance, this is true of the k-shortest paths and k-best spanning trees problems. Suppose that the best solution to the problem is represented by the sequence of edges $e_1, e_2, \ldots$. Then, from this solution, we form a collection of subproblems, one for each of these edges, where the ith subproblem consists of the solutions that are forced to include all the edges in the sequence up to e_{i-1} but that exclude e_i. Every solution to the overall problem, other than the best solution, belongs to exactly one of these subproblems: the one defined by the first difference between the given solution and the best solution. Recursively subdividing each of these subproblems into sub-subproblems, etc., produces a heap-ordered tree whose degree equals the maximum number of edges. Again, applying a selection algorithm in a heap allows the k best solutions to be found. However, because of the higher degree of the min-heap in this technique, the number of subproblems that will be examined is $O(ks)$ and the overall time is $O(ks)$ times the time for finding a single best solution, where s is the maximum number of edges in a single solution.

These two partitioning techniques are reviewed in more detail by Hamacher and Queyranne [95]. They attribute the binary heap partition method to a k-best spanning tree algorithm of Gabow [81]. The multiway partition based only on

the best solution comes from a k-best matching algorithm by Murty [142] and its generalization by Lawler [124].

3 Shortest Paths

Probably the most important and heavily studied of the k-best optimization problems is the problem of finding k shortest paths [2, 4, 9, 10, 14, 15, 23, 31, 39, 41, 43, 66, 70–72, 80, 85–88, 100, 102, 106, 108, 109, 121–125, 127, 133, 138–140, 152–154, 156, 157, 159, 168–170, 172, 173, 182–185, 193].

The k-shortest paths problem includes as special cases finding the k best solutions to problems such as biological sequence alignment [27, 143, 166, 167, 181] or the $(0, 1)$-knapsack problem [187], whose dynamic programming solutions can be expressed as shortest path problems in an associated graph. Many problems of hypothesis generation in natural language processing and speech recognition can also be formulated as k-best optimization problems [20, 35, 38, 45–48, 54, 68, 99, 119, 120, 161, 162, 174]. The Viterbi decoding technique for Markov models, commonly used to model these problems, can also be formulated as a search for a path in an associated graph, with a vertex for each pair of a time step and a Markov state, and the k-best beam search technique used for multiple hypothesis generation in these problems can be interpreted as a special case of a k-shortest-path algorithm. This method can also be used to combine different techniques for language and speech recognition, by using one technique to generate hypotheses and the other technique to rescore them [146].

The many other applications of the k shortest paths problem include reconstruction of metabolic pathways [6] and gene regulation networks [171], motion tracking [17], message routing in communications networks [11, 12, 15, 130, 180], listing close genealogical relationships in highly intermarried pedigrees [66], power line placement [44], vehicle and transportation routing [90, 110, 137, 186], building evacuation planning [113], timing analysis of circuits [8, 189], task scheduling [59, 101], and communications and transportation network design [24, 26, 58, 61, 129], as well as in subroutines for other combinatorial optimization problems [19, 33, 51, 53, 79, 105]

In the most basic version of the problem, the input is a weighted directed graph, with two designated source and destination vertices s and t, and a number k. The goal is to find k different walks (paths allowing repeated vertices) from s to t, with the minimum possible weights. An algorithm by Eppstein [66] achieves optimal asymptotic time complexity: $O(m + n \log n + k)$, constant time per path after a preprocessing stage with the running time of Dijkstra's algorithm for a single shortest path. Eppstein's algorithm begins by computing a tree T of shortest paths to t, and (following Hoffman and Pavley [100]) it represents each of its output

paths by the sequences of *detours* that these paths make: edges that do not belong to T. The length of the path is then the shortest path distance from s to t, plus the sum of the lengths added by each detour. For each vertex v in the graph, Eppstein's algorithm constructs a collection of the detours whose starting vertex is on the path in T from v to t; this collection is represented as a binary heap, using persistent data structure techniques [60] to allow these collections to share substructures with each other to save preprocessing time and storage. The algorithm uses these collections to partition the space of solutions into a collection of subproblems that themselves have the structure of a bounded-degree heap. Each subproblem consists of the paths that start with a given sequence of detours, and that use at least one more detour from a given binary heap of detours. The optimal solution of such a subproblem is the one whose final detour is at the root of the heap, and it has three subproblems with worse solutions as children: two in which the root detour is not used and instead the path uses at least one detour from a child of the root in the binary heap of detours, and one where the root detour is used but is not the last detour, and the next detour comes from the binary heap associated with the endpoint of the root detour.

In a graph with cycles, the k shortest paths may consist of as few as one simple path, together with one or more repetitions of a short cycle starting and ending at one of the vertices of the path. Loops of this type are generally not desired, and the problem is particularly critical when the input is an undirected graph, as converting it to a directed graph by replacing each undirected edge by two directed edge will create many potential loops. Beginning with Clarke, Krikorian, and Rausen [49] researchers have developed algorithms that instead seek the k shortest simple (or loopless) paths from s to t in a network [18, 30, 40, 98, 104, 132, 151, 158, 178, 188]. Yen's algorithm [188] still remains the one with the best asymptotic time performance. It is based on best-solution partitioning and Dijkstra's algorithm; the number of edges in a single solution is at most $n-1$, and the time to find a solution using the Fibonacci-heap variant of Dijkstra's algorithm is $O(m + n \log n)$ (in a graph with m edges and n vertices), so following the general form for best-solution partitioning, the time for this method is $O(kn(m+n \log n)$. A more recent algorithm of Hershberger, Maxal, and Suri [98] is often faster, but is based on a heuristic that can sometimes fail, causing it to fall back to Yen's algorithm and achieve the same performance bound. In the case of undirected graphs, it is possible to find the k shortest simple paths faster, in time $O(k(m + n \log n))$ [89, 115, 117].

Minieka [138] and Fox [72] considered an all-pairs variant of the k-shortest-paths problem in which the goal is to find a separate set of k paths for each pair of vertices in the graph. For this problem, Eppstein's algorithm requires only n copies of the preprocessing stage (one for each destination vertex), after which each path takes constant time to find, so the total time is $O(mn + n^2 \log n + kn^2)$. The shortest path tree in a graph is a tree connecting a given source vertex to all other vertices,

minimizing the sum of the path costs to the other vertices. The k-best version of this problem, seeking the k best trees according to this quality measure, has also been studied [165].

There has also been research on finding a given number of paths between two given terminals that are completely disjoint from each other and minimize a sum of weights [32, 144]. This problem can be solved as a special case of the minimum-cost flow problem; it has a significantly different flavor from k-best enumeration problems, as the choice of one path affects the others and the number of paths that can be selected is much smaller.

4 Spanning Trees

A 1977 paper of Gabow [81] introduced both the problem of finding the k minimum-weight spanning trees of an edge-weighted graph, and the technique of finding a binary hierarchical subdivision of the space of solutions, which he used to solve the problem. In any graph, the best and second-best spanning trees differ only by one edge swap (the removal of one edge from a tree and its replacement by a different edge that reconnects the two subtrees formed by the removal), a property that simplifies the search for a second-best tree as needed for Gabow's partitioning technique. For similar reasons, when k is smaller than the numbers n and m of vertices or edges in the input graph, the graph may be simplified by finding a single minimum spanning tree, computing the best swap that each edge of the graph participates in, removing the edges that are not in the tree and do not participate in the k best swaps, and contracting the edges that are in the tree but do not participate in the k best swaps. For this reason, factors of n and m in the running time of any k-best spanning tree algorithm may be replaced by k when this replacement would be an improvement [64].

The fastest known algorithms for the k best spanning trees problem are based on Gabow's partitioning technique, together with dynamic graph data structures that keep track of the best swap in a network as that network undergoes a sequence of edge insertion and deletion operations [67, 73, 75]. To use this technique, one initializes a fully-persistent best-swap data structure (one in which each update creates a new version of the structure without modifying the existing versions, and in which updates may be applied to any version [60]) and associates its initial version with the root of the subproblem tree. Then, whenever an algorithm for selecting the k best nodes of the subproblem tree generates a new node (a subproblem formed by including or excluding an edge from the allowed solutions) the parent node's version of the data structure is updated (by either increasing or decreasing the weight of the edge to force it to be included or excluded in all solutions) and the updated version of the data structure is associated with the child

node. In this way, the data structure can be used to quickly find the second-best solution for each of the subproblems explored by the algorithm. Based on this method, the k best spanning trees of a graph with n vertices and m edges can be found (in an implicit representation based on sequences of swaps rather than explicitly listing all edges in each tree) in time $O(\mathrm{MST}(m,n)+k\min(n,k)^{1/2})$ where $\mathrm{MST}(m,n)$ denotes the time for finding a single minimum spanning tree [67]. If randomized algorithms are considered, the minimum spanning tree problem can be solved in linear time [114], so the $\mathrm{MST}(m,n)$ term can be replaced by $m+n$. This technique may also be used to find the k best spanning trees of a set of n points in the Euclidean plane, in time $O(n\log n\log k + k\min(n,k)^{1/2})$ [65, 67]; these trees may include pairs of edges that cross each other, but it is also possible to find the k best non-crossing planar spanning trees efficiently [134].

The kth smallest distinct weight of a spanning tree may be obtained by a sequence of at most $k-1$ swaps from any minimum spanning tree, allowing these weights to be generated in polynomial time when k is constant [112,136]. However, when k is an input variable, finding the kth smallest distinct spanning tree weight remains NP-hard [136].

The problem of finding the k best spanning trees has been applied to NP-hard multicriterion optimization problems for spanning trees [50, 96, 176], to point process intensity estimation [97], to the analysis of metabolic pathways [7], to image segmentation [177] and classification [63], to the reconstruction of pedigrees from genetic data [52], to the parsing of natural-language text [1], and to the analysis of electronic circuits [190]. This problem is a special case of finding the k best bases of a matroid, which has also been studied [25,34,95,126]. For additional research on the k smallest spanning trees problem see [16, 25, 116, 128, 175, 179].

5 Other Problems

After paths and spanning trees, probably the next most commonly studied k-best enumeration problem concerns matchings. The problem of finding the k minimum-weight perfect matchings in an edge-weighted graph was introduced by Murty [142]. A later algorithm by Chegireddy and Hamacher [36] solves the problem in time $O(kn^3)$ (where n is the number of vertices in the graph) using the technique of building a binary partition of the solution space. The k-best matchings have been used to find matchings with additional side constraints [13] or with multivariate optimization criteria [163]. They have also been applied for message routing in parallel computing systems [42]. For additional work on this problem see [56, 135, 147, 150].

In natural language processing applications, an important generalization of the k shortest paths problem involves finding the k best parse trees of a context-free

grammar [35, 103, 107, 148, 149, 192]. Other problems whose k-best solutions have been studied include the Chinese postman problem [160], the traveling salesman problem [155], the k best spanning arborescences in a directed network [28], the matroid intersection problem [29, 191], binary search trees and Huffman coding [5], chess strategies [3], the k best integer flows [92, 93, 164], the k smallest cuts in a network [91, 94, 95], and, in probabilistic reasoning, the k best solutions to a graphical model [55, 69, 78, 145].

For many NP-hard optimization problems, where even finding a single best solution is difficult, an approach that has proven very successful is *parameterized complexity*, in which one finds an integer parameter describing the input instance or its solution that is often much smaller than the input size, and designs algorithms whose running time is a fixed polynomial of the input size multiplied by a non-polynomial function of the parameter value. Chen et al. [37] extend this paradigm to k-best problems, showing that, for instance, many NP-hard k-best problems can be solved in polynomial time per solution for graphs of bounded treewidth.

References

[1] Ž. Agić. K-best spanning tree dependency parsing with verb valency lexicon reranking. *Proc. COLING 2012*, 2012.

[2] H. Aljazzar and S. Leue. K^*: a heuristic search algorithm for finding the k shortest paths. *Artificial Intelligence* 175(18):2129–2154, 2011, doi:10.1016/j.artint.2011.07.003.

[3] I. Althöfer. On the K-best mode in computer chess: measuring the similarity of move proposals. *ICCA J.* 20(3):152–165, September 1997.

[4] K. N. Androutsopoulos and K. G. Zografos. Solving the k-shortest path problem with time windows in a time varying network. *Oper. Res. Lett.* 36(6):692–695, 2008, doi:10.1016/j.orl.2008.07.003.

[5] S. Anily and R. Hassin. Ranking the best binary trees. *SIAM J. Comput.* 18(5):882–892, 1989, doi:10.1137/0218060.

[6] M. Arita. Metabolic reconstruction using shortest paths. *Simulation Practice and Theory* 8(1–2):109–125, 2000, doi:10.1016/S0928-4869(00)00006-9.

[7] M. Arita, K. Asai, and T. Nishioka. Graph modeling of metabolism. *Proc. 4th Int. Conf. Computational Molecular Biology (RECOMB '00)*, 2000.

[8] T. Asano and S. Sato. Long path enumeration algorithms for timing verification on large digital systems. *Graph theory with applications to algorithms and computer science (Kalamazoo, Mich., 1984)*, pp. 25–35. Wiley, New York, Wiley-Intersci. Publ., 1985.

[9] J. A. Azevedo, M. E. O. Santos Costa, J. J. E. R. Silvestre Madeira, and E. Q. Vieira Martins. An algorithm for the ranking of shortest paths. *Eur. J. Oper. Res.* 69(1):97–106, 1993, doi:10.1016/0377-2217(93)90095-5.

[10] J. A. Azevedo, J. J. E. R. Silvestre Madeira, E. Q. Vieira Martins, and F. M. A. Pires. A computational improvement for a shortest paths ranking algorithm. *Eur. J. Oper. Res.* 73(1):188–191, 1994, doi:10.1016/0377-2217(94)90162-7.

[11] K. Bala, T. E. Stern, and K. Bala. Algorithms for routing in a linear lightwave network. *Proc. 10th Joint. Conf. IEEE Computer & Communications Socs. (INFOCOM 1991)*, vol. 1, pp. 1–9, 1991, doi:10.1109/INFCOM.1991.147477.

[12] K. Bala, T. E. Stern, D. Simchi-Levi, and K. Bala. Routing in a linear lightwave network. *ACM/IEEE Trans. on Networking* 3(4):459–469, 1995, doi:10.1109/90.413220.

[13] M. O. Ball, U. Derigs, C. Hilbrand, and A. Metz. Matching problems with generalized upper bound side constraints. *Networks* 20(6):703–721, 1990, doi:10.1002/net.3230200602.

[14] H. Beilner. Ein Algorithmus zur Ermittlung k-bester Kantenfolgen zwischen allen Ecken eines Graphen [An algorithm for determining k-best sequences of edges between all vertices of a graph]. *Computing* 10(3):205–220, 1972, doi:10.1007/BF02316908.

[15] R. Bellman and R. Kalaba. On kth best policies. *J. SIAM* 8(4):582–588, 1960, doi:10.1137/0108044.

[16] P. C. Berbert, L. J. R. Freitas Filho, T. A. Almeida, M. B. Carvalho, and A. Yamakami. Artificial immune system to find a set of k-spanning trees with low costs and distinct topologies. *Artificial Immune Systems: 6th International Conference, ICARIS 2007, Santos, Brazil, August 26-29, 2007, Proceedings*, pp. 395–406. Springer, Lecture Notes in Computer Science 4628, 2007, doi:10.1007/978-3-540-73922-7_34.

[17] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using K-shortest paths optimization. *IEEE Trans. Pattern Analysis and Machine Intelligence* 33(9):1806–1819, 2011, doi:10.1109/TPAMI.2011.21.

[18] A. Bernstein. A nearly optimal algorithm for approximating replacement paths and K shortest simple paths in general graphs. *Proc. 21st ACM-SIAM Symp. Discrete Algorithms (SODA '10)*, pp. 742–755. Society for Industrial and Applied Mathematics, 2010.

[19] S. Bespamyatnikh and A. Kelarev. Algorithms for shortest paths and d-cycle problems. *J. Discrete Algorithms* 1(1):1–9, 2003, doi:10.1016/S1570-8667(03)00002-9.

[20] M. Betz and H. Hild. Language models for a spelled letter recognizer. *Proc. Internat. Conf. Acoustics, Speech, and Signal Processing*, vol. 1, pp. 856–859. IEEE, 1995, doi:10.1109/ICASSP.1995.479829.

[21] M. Blum, V. Pratt, R. E. Tarjan, R. W. Floyd, and R. L. Rivest. Time bounds for selection. *J. Comput. System Sci.* 7:448–461, 1973, doi:10.1016/S0022-0000(73)80033-9.

[22] O. Borůvka. O jistém problému minimálním (About a certain minimal problem). *Práce mor. přírodověd. spol. v Brně III* 3:37–58, 1926.

[23] A. W. Brander and M. C. Sinclair. A comparative study of k-shortest path algorithms. *Performance Engineering of Computer and Telecommunications Systems: Proc. UKPEW'95, Liverpool John Moores University, UK, 5–6 September 1995*, pp. 370–379, 1996, doi:10.1007/978-1-4471-1007-1_25.

[24] G. Bruno, G. Ghiani, and G. Improta. A multi-modal approach to the location of a rapid transit line. *Eur. J. Oper. Res.* 104(2):321–332, 1998, doi:10.1016/S0377-2217(97)00187-2.

[25] R. N. Burns and C. E. Haff. A ranking problem in graphs. *Proc. 5th Southeastern Conference on Combinatorics, Graph Theory and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1974)*, pp. 461–470. Utilitas Math., Congressus Numerantium 10, 1974.

[26] M. T. Busche, C. M. Lockhart, and C. Olszewski. Dynamic K-shortest path (DKSP) facility restoration algorithm. *GLOBECOM, Communications: The Global Bridge*, vol. 1, pp. 536–542. IEEE, 1994, doi:10.1109/GLOCOM.1994.513577.

[27] T. H. Byers and M. S. Waterman. Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming. *Oper. Res.* 32(6):1381–1384, 1984, doi:10.1287/opre.32.6.1381.

[28] P. M. Camerini, L. Fratta, and F. Maffioli. The K best spanning arborescences of a network. *Networks* 10(2):91–110, 1980, doi:10.1002/net.3230100202.

[29] P. M. Camerini and H. W. Hamacher. Intersection of two matroids: (condensed) border graphs and ranking. *SIAM J. Discrete Math.* 2(1):16–27, 1989, doi:10.1137/0402002.

[30] W. M. Carlyle and R. K. Wood. Near-shortest and K-shortest simple paths. *Networks* 46(2):98–109, 2005, doi:10.1002/net.20077.

[31] P. Carraresi and C. Sodini. A binary enumeration tree to find K shortest paths. *VII. symposium on operations research, Sections 1–3 (St. Gallen, 1982)*, pp. 177–188. Athenäum/Hain/Hanstein, Methods Oper. Res. 45, 1983.

[32] D. Castanon. Efficient algorithms for finding the K best paths through a trellis. *IEEE Transactions on Aerospace and Electronic Systems* 26(2):405–410, 1990, doi:10.1109/7.53448.

[33] H.-J. Chang and U.-T. Lai. Empirical comparison between two k-shortest path methods for the generalized assignment problem. *J. Inform. Optim. Sci.* 19(2):153–171, 1998, doi:10.1080/02522667.1998.10699369.

[34] B. Chaourar. On the Kth best base of a matroid. *Oper. Res. Lett.* 36(2):239–242, 2008, doi:10.1016/j.orl.2007.05.007.

[35] E. Charniak and M. Johnson. Coarse-to-fine N-best parsing and MaxEnt discriminative reranking. *Proc. 43rd Meeting of Association for Computational Linguistics (ACL '05)*, pp. 173–180, 2005, doi:10.3115/1219840.1219862.

[36] C. R. Chegireddy and H. W. Hamacher. Algorithms for finding K-best perfect matchings. *Discrete Appl. Math.* 18(2):155–165, 1987, doi:10.1016/0166-218X(87)90017-5.

[37] J. Chen, I. A. Kanj, J. Meng, G. Xia, and F. Zhang. Parameterized top-K algorithms. *Theoret. Comput. Sci.* 470:105–119, 2013, doi:10.1016/j.tcs.2012.10.052.

[38] J.-K. Chen and F. K. Soong. An N-best candidates-based discriminative training for speech recognition applications. *Trans. Speech & Audio Processing* 2(1, part 2):206–216, 1994, doi:10.1109/89.260363.

[39] Y. L. Chen. An algorithm for finding the k quickest paths in a network. *Comput. Oper. Res.* 20(1):59–65, 1993, doi:10.1016/0305-0548(93)90096-2.

[40] Y. L. Chen. Finding the k quickest simple paths in a network. *Inform. Process. Lett.* 50(2):89–92, 1994, doi:10.1016/0020-0190(94)00008-5.

[41] Y.-L. Chen and H.-H. Yang. Finding the first K shortest paths in a time-window network. *Comput. Oper. Res.* 31(4):499–513, 2004, doi:10.1016/S0305-0548(02)00230-7.

[42] W.-K. Chiang and R.-J. Chen. Block-switch networks: a cost-effective class of interconnection networks. *Computer Systems Science and Engineering* 12(3):175–185, 1997, `http://ir.lib.nctu.edu.tw/handle/987654321/39344`.

[43] E. I. Chong, S. Maddila, and S. Morley. On finding single-source single-destination k shortest paths. *J. Comput. Inf.* 1(2):40–47, 1995.

[44] F. Choobineh and T. Burgman. Transmission line route selection: An application of K-shortest paths and goal programming. *Trans. Power Apparatus and Systems* PAS-103(11):3253–3259, 1984, doi:10.1109/TPAS.1984.318562.

[45] W. Chou, C.-H. Lee, and B.-H. Juang. Minimum error rate training based on N-best string models. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-93)*, vol. 2, pp. 652–655, 1993, doi:10.1109/ICASSP.1993.319394.

[46] W. Chou, C.-H. Lee, B.-H. Juang, and F. K. Soong. A minimum error rate pattern recognition approach to speech recognition. *Int. J. Pattern Recognition & Artificial Intelligence* 8(1):5–31, 1994, doi:10.1142/S0218001494000024.

[47] W. Chou, T. Matsuoka, B.-H. Juang, and C.-H. Lee. An algorithm of high resolution and efficient multiple string hypothesization for continuous speech recognition using inter-word models. *Proc. Internat. Conf. Acoustics, Speech, and Signal Processing*, vol. 2, pp. II/153–156. IEEE, 1994.

[48] Y.-L. Chow and R. Schwartz. The N-best algorithm: an efficient search procedure for finding top N sentence hypotheses. *Proc. DARPA Speech & Natural Language Worksh.*, pp. 199–202, 1989.

[49] S. Clarke, A. Krikorian, and J. Rausen. Computing the *N* best loopless paths in a network. *J. SIAM* 11(4):1096–1102, 1963, doi:10.1137/0111081.

[50] J. C. N. Clímaco, M. E. Captivo, and M. M. B. Pascoal. On the bicriterion—minimal cost/minimal label—spanning tree problem. *Eur. J. Oper. Res.* 204(2):199–205, 2010, doi:10.1016/j.ejor.2009.10.013.

[51] J. M. Coutinho-Rodrigues, J. C. N. Clímaco, and J. R. Current. An interactive bi-objective shortest path approach: searching for unsupported nondominated solutions. *Comput. Oper. Res.* 26(8):789–798, 1999, doi:10.1016/S0305-0548(98)00094-X.

[52] R. G. Cowell. A simple greedy algorithm for reconstructing pedigrees. *Theor. Population Biol.* 83:55–63, 2013, doi:10.1016/j.tpb.2012.11.002.

[53] J. R. Current, C. S. ReVelle, and J. L. Cohon. The median shortest path problem: a multiobjective approach to analyze cost vs. accessibility in the design of transportation networks. *Transportation Science* 21(3):188–197, 1987, doi:10.1287/trsc.21.3.188.

[54] J.-C. Dai and H.-J. Lee. Parsing with tag information in a probabilistic generalized LR parser. *Proc. Internat. Conf. Chinese Computing*, pp. 33–39. Nat. Univ. Singapore, 1994.

[55] R. Dechter, N. Flerova, and R. Marinescu. Search algorithms for *m* best solutions for graphical models. *Proc. 26th AAAI Conf. Artificial Intelligence*, pp. 1895–1901, 2012, `http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/viewFile/5167/5349`.

[56] U. Derigs and A. Metz. On the construction of the set of *K*-best matchings and their use in solving constrained matching problems. *Combinatorial Optimization (Ankara, 1990)*, pp. 209–223. Springer, NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci. 82, 1992, doi:10.1007/978-3-642-77489-8_11.

[57] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271, 1959, doi:10.1007/BF01386390.

[58] J. M. B. Diogo, J. C. N. Clímaco, P. M. N. Nordeste, and J. M. F. Craveirinha. Dynamic planning model for urban telephone networks and its applications. *IEE Proceedings I (Communications, Speech and Vision)* 136(4):283–290, 1989.

[59] B. M. Dodin. Determining the *K* most critical paths in PERT networks. *Oper. Res.* 32(4):859–877, 1984, doi:10.1287/opre.32.4.859.

[60] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan. Making data structures persistent. *J. Comput. System Sci.* 38(1):86–124, 1989, doi:10.1016/0022-0000(89)90034-2.

[61] D. A. Dunn, W. D. Grover, and M. H. MacGregor. Comparison of *k*-shortest paths and maximum flow routing for network facility restoration. *IEEE J. Selected Areas in Communications* 12(1):88–99, 1994, doi:10.1109/49.265708.

[62] J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *J. Res. Nat. Bur. Standards Sect. B* 69B:125–130, 1965.

[63] S. El Fkihi, M. Daoudi, and D. Aboutajdine. The mixture of K-optimal-spanning-trees based probability approximation: Application to skin detection. *Image and Vision Computing* 26(12):1574–1590, 2008, doi:10.1016/j.imavis.2008.02.003.

[64] D. Eppstein. Finding the k smallest spanning trees. *BIT* 32(2):237–248, 1992, doi:10.1007/BF01994879.

[65] D. Eppstein. Tree-weighted neighbors and geometric k smallest spanning trees. *Internat. J. Comput. Geom. Appl.* 4(2):229–238, 1994, doi:10.1142/S0218195994000136.

[66] D. Eppstein. Finding the k shortest paths. *SIAM J. Comput.* 28(2):652–673, 1998, doi:10.1137/S0097539795290477.

[67] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification—a technique for speeding up dynamic graph algorithms. *J. ACM* 44(5):669–696, 1997, doi:10.1145/265910.265914.

[68] K. Filippova. Multi-sentence compression: Finding shortest paths in word graphs. *Proc. 23rd Internat. Conf. Computational Linguistics (COLING '10)*, pp. 322–330. Association for Computational Linguistics, 2010.

[69] N. Flerova, E. Rollon, and R. Dechter. Bucket and mini-bucket schemes for m best solutions over graphical models. *Proc. 2nd Internat. Conf. Graph Structures for Knowledge Representation and Reasoning (GKR'11)*, pp. 91–118. Springer, Lecture Notes in Computer Science 7205, 2012, doi:10.1007/978-3-642-29449-5_4.

[70] B. L. Fox. Calculating kth shortest paths. *INFOR–Canad. J. Op. Res. & Inf. Proc.* 11:66–70, 1973.

[71] B. L. Fox. k-th shortest paths and applications to the probabilistic networks. *Bull. Operations Research Soc. of America* 23:B263, 1975.

[72] B. L. Fox. More on kth shortest paths. *Commun. ACM* 18(5):279, 1975, doi:10.1145/360762.360803.

[73] G. N. Frederickson. Data structures for on-line updating of minimum spanning trees, with applications. *SIAM J. Comput.* 14(4):781–798, 1985, doi:10.1137/0214055.

[74] G. N. Frederickson. An optimal algorithm for selection in a min-heap. *Inform. and Comput.* 104(2):197–214, 1993, doi:10.1006/inco.1993.1030.

[75] G. N. Frederickson. Ambivalent data structures for dynamic 2-edge-connectivity and k smallest spanning trees. *SIAM J. Comput.* 26(2):484–538, 1997, doi:10.1137/S0097539792226825.

[76] G. N. Frederickson and D. B. Johnson. The complexity of selection and ranking in $X + Y$ and matrices with sorted columns. *J. Comput. System Sci.* 24(2):197–208, 1982, doi:10.1016/0022-0000(82)90048-4.

[77] G. N. Frederickson and D. B. Johnson. Generalized selection and ranking: sorted matrices. *SIAM J. Comput.* 13(1):14–30, 1984, doi:10.1137/0213002.

[78] A. Fromer, M. and Globerson. An LP view of the M-best MAP problem. *Advances in Neural Information Processing Systems* 22:567–575, 2009, `http://papers.nips.cc/paper/3745-an-lp-view-of-the-m-best-map-problem`.

[79] L. Fu and L. R. Rilett. Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Res. B* 32(7):499–516, 1998, doi:10.1016/S0191-2615(98)00016-2.

[80] A. Fuchs. 1–n Verfahren zur Bestimmung von k-kürzesten Wegen [1–n method for determining k-shortest paths]. *Operations Research Proceedings 1984 (St. Gallen, 1984)*, pp. 421–428. Springer, 1985, doi:10.1007/978-3-642-70457-4_108.

[81] H. N. Gabow. Two algorithms for generating weighted spanning trees in order. *SIAM J. Comput.* 6(1):139–150, 1977, doi:10.1137/0206011.

[82] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM* 38(4):815–853, 1991, doi:10.1145/115234.115366.

[83] Z. Galil, S. Micali, and H. N. Gabow. An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Comput.* 15(1):120–130, 1986, doi:10.1137/0215009.

[84] P. N. Gilbert. New results in planar triangulations. Master's thesis, Coordinated Science Lab., Univ. of Illinois, Urbana, 1979.

[85] N. Gravin and N. Chen. A note on k-shortest paths problem. *J. Graph Theory* 67(1):34–37, 2011, doi:10.1002/jgt.20510.

[86] F. Guerriero and R. Musmanno. Parallel asynchronous algorithms for the K shortest paths problem. *J. Optim. Theory Appl.* 104(1):91–108, 2000, doi:10.1023/A:1004676705907.

[87] F. Guerriero, R. Musmanno, V. Lacagnina, and A. Pecorella. A class of label-correcting methods for the K shortest paths problem. *Oper. Res.* 49(3):423–429, 2001, doi:10.1287/opre.49.3.423.11217.

[88] M. Günther, J. Schuster, and M. Siegle. Symbolic calculation of K-shortest paths and related measures with the stochastic process algebra tool CASPA. *Proc. 1st Worksh. Dynamic Aspects in Dependability Models for Fault-Tolerant Systems (DYADEM-FTS '10)*, pp. 13–18. ACM, 2010, doi:10.1145/1772630.1772635.

[89] E. Hadjiconstantinou and N. Christofides. An efficient implementation of an algorithm for finding K shortest simple paths. *Networks* 34(2):88–101, 1999, doi:10.1002/(SICI)1097-0037(199909)34:2<88::AID-NET2>3.3.CO;2-T.

[90] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. *Ann. Oper. Res.* 61:21–43, 1995, doi:10.1007/BF02098280.

[91] H. W. Hamacher. An $(K \cdot n^4)$ algorithm for finding the k best cuts in a network. *Oper. Res. Lett.* 1(5):186–189, 1982, doi:10.1016/0167-6377(82)90037-2.

[92] H. W. Hamacher. A note on *K* best network flows. *Ann. Oper. Res.* 57:65–72, 1995, doi:10.1007/BF02099691.

[93] H. W. Hamacher and C. Hüsselmann. Ranking approach to max-ordering combinatorial optimization and network flows. *Beiträge zur angewandten Analysis und Informatik*, pp. 97–111. Shaker, 1994.

[94] H. W. Hamacher, J.-C. Picard, and M. Queyranne. On finding the *K* best cuts in a network. *Oper. Res. Lett.* 2(6):303–305, 1984, doi:10.1016/0167-6377(84)90083-X.

[95] H. W. Hamacher and M. Queyranne. *K* best solutions to combinatorial optimization problems. *Ann. Oper. Res.* 4(1–4):123–143, 1985, doi:10.1007/BF02022039.

[96] H. W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Ann. Oper. Res.* 52:209–230, 1994, doi:10.1007/BF02032304.

[97] A. O. Hero and O. Michel. Robust estimation of point process intensity features using *k*-minimal spanning trees. *Proc. Internat. Symp. Information Theory*, p. 74. IEEE, 1997, doi:10.1109/ISIT.1997.612989.

[98] J. Hershberger, M. Maxel, and S. Suri. Finding the *k* shortest simple paths: a new algorithm and its implementation. *ACM Trans. Algorithms* 3(4):A45, 2007, doi:10.1145/1290672.1290682.

[99] T. Hisamitsu and Y. Nitta. A generalized algorithm for Japanese morphological analysis and a comparative evaluation of some heuristics. *Systems & Computers in Japan* 26(1):73–87, 1995, doi:10.1002/scj.4690260107.

[100] W. Hoffman and R. Pavley. A method for the solution of the *N*th best path problem. *J. ACM* 6(4):506–514, 1959, doi:10.1145/320998.321004.

[101] G. J. Horne. Finding the *K* least cost paths in an acyclic activity network. *J. Oper. Res. Soc.* 31(5):443–448, 1980, `http://www.jstor.org/stable/2581195`.

[102] T.-Y. Hu, H. S. Mahmassani, and A. K. Ziliaskopoulos. Implementation and testing of a *K*-shortest path algorithm in a vector and parallel processing environment. *Proc. Conf. Computer Science & Operations Research: New Developments & their Interfaces*, 1992.

[103] L. Huang and D. Chiang. Better *k*-best parsing. *Proc. 9th Internat. Worksh. Parsing Technology (IWPT '05)*, pp. 53–64. Association for Computational Linguistics, 2005, `http://www.aclweb.org/anthology/W05-1506.pdf`.

[104] H. Ishii. A new method finding the *K*th best path in a graph. *J. Oper. Res. Soc. Japan* 21(4):469–476, 1978.

[105] Z. Jia and P. Varaiya. Heuristic methods for delay constrained least cost routing using *k*-shortest-paths. *IEEE Trans. Automat. Control* 51(4):707–712, 2006, doi:10.1109/TAC.2006.872827.

[106] V. M. Jiménez and A. Marzal. Computing the *K* shortest paths: A new algorithm and an experimental comparison. *Algorithm Engineering: 3rd International Workshop, WAE'99 London, UK, July 19–21, 1999, Proceedings*, pp. 15–29. Springer, Lecture Notes in Computer Science 1668, 1999, doi:10.1007/3-540-48318-7_4.

[107] V. M. Jiménez and A. Marzal. Computation of the N best parse trees for weighted and stochastic context-free grammars. *Proc. Joint IAPR Internat. Workshops on Advances in Pattern Recognition*, pp. 183–192. Springer, Lecture Notes in Computer Science 1876, 2000, doi:10.1007/3-540-44522-6_19.

[108] V. M. Jiménez and A. Marzal. A lazy version of Eppstein's K shortest paths algorithm. *Experimental and Efficient Algorithms: Second International Workshop, WEA 2003, Ascona, Switzerland, May 26–28, 2003, Proceedings*, pp. 179–190. Springer, Lecture Notes in Computer Science 2647, 2003, doi:10.1007/3-540-44867-5_14.

[109] L.-M. Jin and S.-P. Chan. An electrical method for finding suboptimal routes. *Int. Symp. Circuits and Systems*, vol. 2, pp. 935–938. IEEE, 1989, doi:10.1109/ISCAS.1989.100505.

[110] W. Jin, S. Chen, and H. Jiang. Finding the K shortest paths in a time-schedule network with constraints on arcs. *Comput. Oper. Res.* 40(12):2975–2982, 2013, doi:10.1016/j.cor.2013.07.005.

[111] D. B. Johnson and S. D. Kashdan. Lower bounds for selection in $X + Y$ and other multisets. *J. ACM* 25(4):556–570, 1978, doi:10.1145/322092.322097.

[112] M. Kano. Maximum and kth maximal spanning trees of a weighted graph. *Combinatorica* 7(2):205–214, 1987, doi:10.1007/BF02579450.

[113] C. J. Karbowicz and J. M. Smith. A K-shortest paths routing heuristic for stochastic network evacuation models. *Engineering Optimization* 7(4):253–280, 1984, doi:10.1080/03052158408960642.

[114] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM* 42(2):321–328, 1995, doi:10.1145/201019.201022.

[115] N. Katoh, T. Ibaraki, and H. Mine. An $O(Kn^2)$ algorithm for K shortest simple paths in an undirected graph with nonnegative arc length. *Electron. Comm. Japan* 61(12):1–8 (1980), 1978.

[116] N. Katoh, T. Ibaraki, and H. Mine. An algorithm for finding K minimum spanning trees. *SIAM J. Comput.* 10(2):247–255, 1981, doi:10.1137/0210017.

[117] N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for K shortest simple paths. *Networks* 12(4):411–427, 1982, doi:10.1002/net.3230120406.

[118] G. T. Klincsek. Minimal triangulations of polygonal domains. *Combinatorics 79*, pp. 121–123. Elsevier, Ann. Discrete Math. 9, 1980, doi:10.1016/S0167-5060(08)70044-X.

[119] K. Knight and J. Graehl. Machine transliteration. *Comput. Linguist.* 24(4):599–612, 1998.

[120] K. Knight and V. Hatzivassiloglou. Two-level, many-paths generation. *Proc. Conf. Assoc. for Computational Linguistics*, pp. 252–260, 1995.

[121] S. Kundu. An incremental algorithm for identification of longest (shortest) paths. *Integration* 17(1):25–31, 1994, doi:10.1016/0167-9260(94)90018-3.

[122] P. I. Lalov and T. G. Vasil'eva. Transformation of an oriented graph for finding the *k* shortest paths. *Godishnik Vissh. Uchebn. Zaved. Prilozhna Mat.* 23(4):189–194, 1987.

[123] A. G. Law and A. Rezazadeh. Computing the *K*-shortest paths, under nonnegative weighting. *Twenty-second Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, MB, 1992)*, pp. 277–280. Utilitas Math., Congressus Numerantium 92, 1993.

[124] E. L. Lawler. A procedure for computing the *K* best solutions to discrete optimization problems and its application to the shortest path problem. *Management Sci.* 18(7):401–405, 1972, doi:10.1287/mnsc.18.7.401.

[125] E. L. Lawler. Comment on computing the *k* shortest paths in a graph. *Commun. ACM* 20(8):603–604, 1977, doi:10.1145/359763.359804.

[126] M. Leclerc and F. Rendl. *k*-best constrained bases of a matroid. *Z. Oper. Res.* 34(2):79–89, 1990, doi:10.1007/BF01415971.

[127] G. Liu and K. G. Ramakrishnan. A*Prune: an algorithm for finding *K* shortest paths subject to multiple constraints. *Proc. 20th Joint Conf. IEEE Computer & Communications Socs. (INFOCOM 2001)*, vol. 2, pp. 743–749, 2001, doi:10.1109/INFCOM.2001.916263.

[128] J. Ma, K. Iwama, and Q.-P. Gu. A parallel algorithm for *k*-minimum spanning trees. *Proc. 2nd AIZU Int. Symp. Parallel Algorithms/Architecture Synthesis*, pp. 384–388. IEEE, 1997, doi:10.1109/AISPAS.1997.581705.

[129] M. H. MacGregor and W. D. Grover. Optimized *k*-shortest-paths algorithm for facility restoration. *Software: Practice and Experience* 24(9):823–834, 1994, doi:10.1002/spe.4380240904.

[130] S. W. Mao, H. G. Zhang, C. C. Huang, and W. Q. Wu. A new fault-tolerance mechanism in communications based on the *K* shortest path algorithm. *J. Wuhan Univ. Natur. Sci. Ed.* 59(6):534–538, 2013.

[131] C. Martínez and S. Roura. Optimal sampling strategies in quicksort and quickselect. *SIAM J. Comput.* 31(3):683–705, 2001, doi:10.1137/S0097539700382108.

[132] E. Q. V. Martins and M. M. B. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. *4OR* 1(2):121–133, 2003, doi:10.1007/s10288-002-0010-2.

[133] H. Maruyama. A new *k*th-shortest path algorithm. *IEICE Trans. Information & Systems* E76-D(3):388–389, 1993.

[134] A. Marzetta and J. Nievergelt. Enumerating the *k* best plane spanning trees. *Comput. Geom. Theory and Appl.* 18(1):55–64, 2001, doi:10.1016/S0925-7721(00)00029-8.

[135] T. Matsui, A. Tamura, and Y. Ikebe. Algorithms for finding a *K*th best valued assignment. *Discrete Appl. Math.* 50(3):283–296, 1994, doi:10.1016/0166-218X(92)00175-L.

[136] E. W. Mayr and C. G. Plaxton. On the spanning trees of weighted graphs. *Combinatorica* 12(4):433–447, 1992, doi:10.1007/BF01305236.

[137] S.-P. Miaou and S.-M. Chin. Computing *k*-shortest path for nuclear spent fuel highway transportation. *Eur. J. Oper. Res.* 53(1):64–80, 1991, doi:10.1016/0377-2217(91)90093-B.

[138] E. Minieka. On computing sets of shortest paths in a graph. *Commun. ACM* 17(6):351–353, 1974, doi:10.1145/355616.364037.

[139] E. Minieka. The *K*-th shortest path problem. *Bull. Operations Research Soc. of America* 23:B/116, 1975.

[140] E. Minieka and D. R. Shier. A note on an algebra for the *k* best routes in a network. *IMA J. Appl. Math.* 11(2):145–149, 1973, doi:10.1093/imamat/11.2.145.

[141] W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *J. ACM* 55(2):A11, 2008, doi:10.1145/1346330.1346336, arXiv:cs.CG/0601002.

[142] K. G. Murty. Letter to the editor—An algorithm for ranking all the assignments in order of increasing cost. *Oper. Res.* 16(3):682–687, 1968, doi:10.1287/opre.16.3.682.

[143] D. Naor and D. Brutlag. On near-optimal alignments of biological sequences. *J. Computational Biology* 1(4):349–366, 1994, doi:10.1089/cmb.1994.1.349.

[144] S. Nikolopoulos, A. Pitsillides, and D. Tipper. Addressing network survivability issues by finding the *K*-best paths through a trellis graph. *Proc. 16th Joint Conf. IEEE Computer & Communications Socs. (INFOCOM 1997)*, vol. 1, pp. 370–377, 1997, doi:10.1109/INFCOM.1997.635161.

[145] D. Nilsson. An efficient algorithm for finding the *M* most probable configurations in probabilistic expert systems. *Statistics and Computing* 8(2):159–173, 1998, doi:10.1023/A:1008990218483.

[146] M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. R. Rohlicek. Integration of diverse recognition methodologies through reevaluation of N-best sentence hypotheses. *Proc. Worksh. Speech and Natural Language (HLT '91)*, pp. 83–87. Association for Computational Linguistics, 1991, doi:10.3115/112405.112416.

[147] M. M. B. Pascoal, M. E. Captivo, and J. C. N. Clímaco. A note on a new variant of Murty's ranking assignments algorithm. *4OR* 1(3):243–255, 2003, doi:10.1007/s10288-003-0021-7.

[148] A. Pauls and D. Klein. *K*-best A* parsing. *Proc. Joint Conf. 47th Annual Meeting of the ACL & 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL '09)*, vol. 2, pp. 958–966. Association for Computational Linguistics, 2009.

[149] A. Pauls, D. Klein, and C. Quirk. Top-down *k*-best A* Parsing. *Proc. ACL 2010 Conference Short Papers (ACLShort '10)*, pp. 200–204. Association for Computational Linguistics, 2010.

[150] C. R. Pedersen, L. R. Nielsen, and K. A. Andersen. An algorithm for ranking assignments using reoptimization. *Comput. Oper. Res.* 35(11):3714–3726, 2008, doi:10.1016/j.cor.2007.04.008.

[151] A. Perko. Implementation of algorithms for *k* shortest loopless paths. *Networks* 16(2):149–160, 1986, doi:10.1002/net.3230160204.

[152] M. Pollack. Letter to the Editor—The *k*th best route through a network. *Oper. Res.* 9(4):578–580, 1961, doi:10.1287/opre.9.4.578.

[153] M. Pollack. Solutions of the *k*th best route through a network—a review. *J. Math. Anal. and Appl.* 3(3):547–559, 1961, doi:10.1016/0022-247X(61)90076-2.

[154] M. Pollack. Shortest path solutions of the *k*th best route problems. *Bull. Operations Research Soc. of America*, 1969.

[155] E. S. van der Poort, M. Libura, G. Sierksma, and J. A. A. van der Veen. Solving the *k*-best traveling salesman problem. *Comput. Oper. Res.* 26(4):409–425, 1999, doi:10.1016/S0305-0548(98)00070-7.

[156] E. de Queirós Vieira Martins, M. M. B. Pascoal, and J. L. E. Dos Santos. Deviation algorithms for ranking shortest paths. *Internat. J. Found. Comput. Sci.* 10(3):247–261, 1999, doi:10.1142/S0129054199000186.

[157] K. A. Rink, E. Y. Rodin, and V. Sundarapandian. A simplification of the double-sweep algorithm to solve the *k*-shortest path problem. *Appl. Math. Lett.* 13(8):77–85, 2000, doi:10.1016/S0893-9659(00)00099-9.

[158] L. Roditty. On the *k* shortest simple paths problem in weighted directed graphs. *SIAM J. Comput.* 39(6):2363–2376, 2010, doi:10.1137/080730950.

[159] E. Ruppert. Finding the *k* shortest paths in parallel. *Algorithmica* 28(2):242–254, 2000, doi:10.1007/s004530010038.

[160] Y. Saruwatari and T. Matsui. A note on *K*-best solutions to the Chinese postman problem. *SIAM J. Optim.* 3(4):726–733, 1993, doi:10.1137/0803037.

[161] R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple (*N*-best) sentence hypotheses. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-91)*, pp. 701–704, 1991, doi:10.1109/ICASSP.1991.150436.

[162] R. Schwartz and Y.-L. Chow. The *N*-best algorithms: an efficient and exact procedure for finding the N most likely sentence hypotheses. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-90)*, pp. 81–84, 1990, doi:10.1109/ICASSP.1990.115542.

[163] D. Schweigert. Vector-weighted matchings. *Combinatorics Advances (Tehran, 1994)*, pp. 267–276. Kluwer Acad. Publ., Mathematics and Its Applications 329, 1995, doi:10.1007/978-1-4613-3554-2_19.

[164] A. Sedeño-Noda and J. J. Espino-Martín. On the K best integer network flows. *Comput. Oper. Res.* 40(2):616–626, 2013, doi:10.1016/j.cor.2012.08.014.

[165] A. Sedeño-Noda and C. González-Martín. On the K-shortest path trees problem. *Eur. J. Oper. Res.* 202(3):628–635, 2010, doi:10.1016/j.ejor.2009.06.017.

[166] T. Shibuya and H. Imai. Enumerating suboptimal alignments of multiple biological sequences efficiently. *Proc. 2nd Pacific Symp. Biocomputing*, pp. 409–420, January 1997.

[167] T. Shibuya and H. Imai. New flexible approaches for multiple sequence alignment. *J. Computational Biology* 4(3):385–413, 1997, doi:10.1089/cmb.1997.4.385.

[168] D. R. Shier. Computational experience with an algorithm for finding the k shortest paths in a network. *J. Res. Nat. Bur. Standards Sect. B* 78B:139–165, 1974.

[169] D. R. Shier. Iterative methods for determining the k shortest paths in a network. *Networks* 6(3):205–229, 1976, doi:10.1002/net.3230060303.

[170] D. R. Shier. On algorithms for finding the k shortest paths in a network. *Networks* 9(3):195–214, 1979, doi:10.1002/net.3230090303.

[171] Y.-K. Shih and S. Parthasarathy. A single source K-shortest paths algorithm to infer regulatory pathways in a gene network. *Bioinformatics* 28(12):i49–i58, 2012, doi:10.1093/bioinformatics/bts212.

[172] C. C. Skicism and B. L. Golden. Solving k-shortest and constrained shortest path problems efficiently. *Network Optimization and Applications*, pp. 249–282, Ann. Oper. Res. 20, 1989, doi:10.1007/BF02216932.

[173] C. C. Skiscim and B. L. Golden. Computing k-shortest path lengths in Euclidean networks. *Networks* 17(3):341–352, 1987, doi:10.1002/net.3230170308.

[174] F. K. Soong and E.-F. Huang. A tree-trellis based fast search for finding the N-best sentence hypotheses in continuous speech recognition. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-91)*, pp. 705–708, 1991, doi:10.1109/ICASSP.1991.150437.

[175] K. Sörensen and G. K. Janssens. An algorithm to generate all spanning trees of a graph in order of increasing cost. *Pesqui. Oper.* 25(2):219–229, 2005, doi:10.1590/S0101-74382005000200004.

[176] S. Steiner and T. Radzik. Computing all efficient solutions of the biobjective minimum spanning tree problem. *Comput. Oper. Res.* 35(1):198–211, 2008, doi:10.1016/j.cor.2006.02.023.

[177] C. Straehle, S. Peter, U. Köthe, and F. A. Hamprecht. K-smallest spanning tree segmentations. *Pattern Recognition: 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013, Proceedings*, pp. 375–384. Springer, Lecture Notes in Computer Science 8142, 2013, doi:10.1007/978-3-642-40602-7_40.

[178] K. Sugimoto and N. Katoh. An algorithm for finding k shortest loopless paths in a directed network. *Trans. Information Processing Soc. Japan* 26:356–364, 1985.

[179] C. S. Tang and W. F. Liang. A parallel algorithm for finding K minimum spanning trees. *J. China Univ. Sci. Tech.* 20(4):464–471, 1990.

[180] D. M. Topkis. A k shortest path algorithm for adaptive routing in communications networks. *IEEE Trans. Comm.* 36(7):855–859, 1988, doi:10.1109/26.2815.

[181] M. S. Waterman. Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proc. Natl. Acad. Sci. U.S.A.* 80(10):3123–3124, 1983, doi:10.1073/pnas.80.10.3123.

[182] M. M. Weigand. Algorithmus 20: Ein leistungsfähiger Algorithmus zur Bestimmung von K-kürzesten Wegen in einem Graphen [Algorithm 20: An efficient algorithm for the determination of k-shortest paths in a graph]. *Computing* 12(3):273–283, 1974, doi:10.1007/BF02293111.

[183] M. M. Weigand. Algorithmus 27: Ein neuer Algorithmus zur Bestimmung von k-kürzesten Wegen in einem Graphen [Algorithm 27: A new algorithm for the determination of k-shortest paths in a graph]. *Computing* 16(1–2):139–151, 1976, doi:10.1007/BF02241985.

[184] A. Weintraub. The shortest and the K-shortest routes as assignment problems. *Networks* 3(1):61–73, 1973, doi:10.1002/net.3230030105.

[185] A. Wongseelashote. An algebra for determining all path-values in a network with application to K-shortest-paths problems. *Networks* 6(4):307–334, 1976, doi:10.1002/net.3230060403.

[186] W. Xu, S. He, R. Song, and S. S. Chaudhry. Finding the K shortest paths in a schedule-based transit network. *Comput. Oper. Res.* 39(8):1812–1826, 2012, doi:10.1016/j.cor.2010.02.005.

[187] H. H. Yanasse, N. Y. Soma, and N. Maculan. An algorithm for determining the K-best solutions of the one-dimensional knapsack problem. *Pesqui. Oper.* 20(1):117–134, 2000, doi:10.1590/S0101-74382000000100011.

[188] J. Y. Yen. Finding the K shortest loopless paths in a network. *Management Sci.* 17:712–716, 1971, `http://www.jstor.org/stable/2629312`.

[189] S. H. Yen, D. H. Du, and S. Ghanta. Efficient algorithms for extracting the K most critical paths in timing analysis. *Proc. 26th ACM/IEEE Design Automation Conf. (DAC '89)*, pp. 649–654. ACM, 1989, doi:10.1145/74382.74497.

[190] Q. Yu and C. Sechen. Generation of colour-constrained spanning trees with application in symbolic circuit analysis. *Int. J. Circuit Theory and Appl.* 24(5):597–603, 1996, doi:10.1002/(SICI)1097-007X(199609/10)24:5<597::AID-CTA938>3.0.CO;2-Q.

[191] Q. Yu and C. Sechen. Efficient approximation of symbolic network functions using matroid intersection algorithms. *IEEE Trans. Computer-Aided Design of Integrated Circuits & Systems* 16(10):1073–1081, 1997, doi:10.1109/43.662671.

[192] Y. Zhang, S. Oepen, and J. Carroll. Efficiency in unification-based n-best parsing. *Proc. 10th Internat. Conf. Parsing Technologie (IWPT '07)*, pp. 48–59. Association for Computational Linguistics, 2007, `http://www.aclweb.org/anthology/W07-2207`.

[193] N. J. van der Zijpp and S. Fiorenzo Catalano. Path enumeration by finding the constrained K-shortest paths. *Transportation Res. B* 39(6):545–563, 2005, doi:10.1016/j.trb.2004.07.004.

THE COMPUTATIONAL COMPLEXITY COLUMN

BY

VIKRAMAN ARVIND

Institute of Mathematical Sciences, CIT Campus, Taramani
Chennai 600113, India
arvind@imsc.res.in
http://www.imsc.res.in/~arvind

We revisit *robust machines* and *helping oracles* introduced by Uwe Schöning [23] three decades ago. A *robust* oracle machine always accepts the same language, regardless of the oracle. An oracle A is said to *help* a robust machine if oracle access to A "speeds up" the machine and makes it polynomial-time bounded. Robust machines with helping oracles actually models interactive computation, and can be seen as a precursor to interactive proofs. We discuss these connections and point out how robust oracle machines relate to some recently defined classes like oblivious NP and oblivious MA [15].

As we keep pace with new developments in the field, it is also worthwhile recalling some of the older ideas and results. Robust machines and helping oracles are a nice example.

BEATCS no 115

Robust Oracle Machines revisited

V. Arvind *

1 Introduction

Let M be an *oracle Turing machine* and let $L(M^A) \subseteq \Sigma^*$ denote the language accepted by the machine M with oracle A, where Σ is a fixed alphabet and inputs are encoded as strings in Σ^*.

Definition 1. [23]

- *An oracle machine M is* robust *if $L(M^A) = L(M^\emptyset)$ for every oracle $A \subseteq \Sigma^*$.*
- *An oracle set $A \subseteq \Sigma^*$* helps *a robust oracle machine M if M^A is polynomial-time bounded on all inputs.*

What is the motivation for studying robust machines and helping oracles? We quote excerpts from the article [23, Section 1]:

> *Typical algorithms for computationally hard problems like NP-complete problems usually involve backtrack search ... consider the situation that the algorithm is allowed to query an oracle during the computation to receive information that might lead to faster search for the solution. This situation can be thought of as some kind of man-machine interaction. ... This model only makes sense if we do not allow the algorithm to rely on the oracle information such that changing the oracle ... would result in changing the final outcome of the algorithm.*

This is an entirely new take on oracle machine computations, quite different from relativization results that were prevalent in structural complexity theory in the 1980's. It provides an algorithmic motivation for the idea of *interactive computation*, and it is perhaps the first time a notion of interactive computation was formulated and studied. The notion of interactive proof systems [16], discovered soon after, has its motivation in cryptography.

Once we have Definition 1, the natural question is which languages have robust oracle machines with oracles that help? We can define the following complexity class [23]:

*Institute of Mathematical Sciences, Chennai, India arvind@imsc.res.in

$$\mathsf{P}_{\mathsf{help}} = \{L^A(M) \mid M \text{ is a robust deterministic Turing machine and } A \text{ helps } M\}.$$

It turns out that $\mathsf{P}_{\mathsf{help}}$ coincides with $\mathsf{NP} \cap \mathsf{coNP}$.

Theorem 2. [23] $\mathsf{P}_{\mathsf{help}} = \mathsf{NP} \cap \mathsf{coNP}$.

Proof Sketch. For the forward inclusion notice that it suffices to show $\mathsf{P}_{\mathsf{help}}$ is contained in NP since $\mathsf{P}_{\mathsf{help}}$ is closed under complements. Suppose $L \in \mathsf{P}_{\mathsf{help}}$. Then $L = L(M^A)$ where M is a robust machine and A is a helping oracle. An NP machine can be easily obtained from M by nondeterministically guessing the answers of oracle A and aborting all computation paths at a suitable polynomial length. Since M is robust this NP machine will not accept $x \notin L$. On the other hand, if $x \in L$ then the path of correct guesses to the queries to A will result in an accepting computation of polynomial length.

For the reverse inclusion, suppose $L \in \mathsf{NP} \cap \mathsf{coNP}$. Let N be an NP machine for L and N' for $\overline{L}$. We design an oracle

$$A = \{\langle x, y\rangle \mid \text{ if } x \in L \text{ then } y \text{ is a prefix of an accepting path of } N(x) \text{ and if } x \notin L \text{ then } y \text{ is a prefix of an accepting path of } N'(x)\}.$$

The robust machine M is a polynomial-time procedure that on input x prefix searches for accepting paths of either $N(x)$ or $N'(x)$ by making oracle queries $\langle x, y\rangle$. In the end, if neither $N(x)$ nor $N'(x)$ accepts, M does a brute-force search to solve x.

□

Remark 3. *Schöning's work on robust machines sparked a lot of interest and related research. An (incomplete) list of papers on this and related topics is [18, 6, 24, 7, 17, 10, 14, 21, 3]. In this article, we consider only some aspects of the topic.*

1.1 *A Proof System Definition*

We can give an equivalent definition of $\mathsf{P}_{\mathsf{help}}$ that is based on the idea of proof systems.

A language L is in $\mathsf{P}_{\mathsf{help}}$ if and only if there is a *polynomial time bounded* oracle Turing machine M such that for any input x and any oracle A the output $M^(x)$ is in $\{Acc, Rej, ?\}$ and the following conditions of "soundness" and "completeness" hold:

Soundness: For every $x \in \Sigma^*$ and every oracle A:

$$M^A(x) = Acc \implies x \in L$$
$$M^A(x) = Rej \implies x \notin L$$

Completeness: There is an oracle $\hat{A}$ such that $M^{\hat{A}}(x) \in \{Acc, Rej\}$ for every input x.

The equivalence with the original definition is straightforward. In this course of this article we will see that robustn machines and helping oracles are, in fact, quite closely connected to interactive proof systems.

2 One-sided help

The notion of one-sided helping oracles was introduced and studied by Ker-I Ko [18] as a generalization of helping.

Definition 4. [18] *A language L is in $\mathsf{P}_{1-\mathsf{help}}$ if there is a robust oracle machine M accepting L, an oracle A and a polynomial time bound $p(n)$ such that for all $x \in \Sigma^*$: if $x \in L$ then $M^(x)$ accepts x in $p(|x|)$ time.*

Notice that the helping oracle A helps only accepts "yes" instances in polynomial time. Indeed, it is easy to see from the definition that the following holds.

Theorem 5. [18] $\mathsf{P}_{1-\mathsf{help}} = \mathsf{NP}$.

In his paper, Ko also studied the subclass $\mathsf{P}_{1-\mathsf{help}}[C]$ of $\mathsf{P}_{1-\mathsf{help}}$ when the helping oracle is restricted to some language class C. For instance $\mathsf{P}_{1-\mathsf{help}}[\mathsf{NP}]$ is the class of languages in $\mathsf{P}_{1-\mathsf{help}}$ that have helping oracles in NP. It is easy to see that this is no restriction as $\mathsf{P}_{1-\mathsf{help}} = \mathsf{P}_{1-\mathsf{help}}[\mathsf{NP}]$.

It is interesting to investigate the power of helping oracles that are known to be not NP-hard. The class $\mathsf{P}_{1-\mathsf{help}}[A]$ for such oracles A would yield uniform subclasses of NP that do not contain NP-complete sets. We note two further results from Ko's article here. He showed that the class $\mathsf{P}_{1-\mathsf{help}}[\mathrm{Sparse}]$, where Sparse is the collection of all polynomially sparse languages, does not contain NP-complete languages unless the polynomial-time hierarchy collapses to the second level. He also showed that log^*-sparse sets are "no helpers", in the sense that they are powerless as helping oracles to robust machines. A language S is said to be log^*-sparse if the number of strings in S of length between n and 2^n is bounded by a fixed constant k for every $n > 0$.

Theorem 6. [18]

- *The class* $\mathsf{P}_{1-\mathsf{help}}[\mathrm{Sparse}]$ *does not contain* NP*-complete languages unless* $\mathsf{PH} = \Sigma_2^p$.
- $\mathsf{P}_{1-\mathsf{help}}[S] = \mathsf{P}$ *for every log^*-sparse language S.*

It is also interesting to consider the power of helping oracles that come from complexity classes not known to contain NP. Recall that a language L is in $\mathsf{Mod}_k\mathsf{P}$ if there is an NP machine M such that

$$x \in L \iff acc_M(x) \not\equiv 0 (mod\ k),$$

where $acc_M(x)$ denotes the number of accepting paths of M on input x.

Ogihara [21] has examined the classes $\mathsf{P}_{1-\mathsf{help}}[\mathsf{Mod}_k\mathsf{P}]$, for prime k, and obtained the following curious result.

Theorem 7. [21] *A language L is in $\mathsf{P}_{1-\mathsf{help}}[\mathsf{Mod}_k\mathsf{P}]$, k prime, if and only if there is an NP machine M that accepts L such that $x \in L \iff acc_M(x) \not\equiv 0 (mod\ k)$.*

Remark 8. *In general, it would be interesting to further investigate the classes $\mathsf{P}_{1-\mathsf{help}}[C]$ and $\mathsf{P}_{\mathsf{help}}[C]$ for different language classes C.*

3 Interactive proof systems

In this section we will start with observations from [24] on *probabilistic robust machines*. Then we will discuss its equivalence with MIP (where MIP stands for the class of languages with multi-prover interactive protocols). Also, compare with the definition of IP (the class of languages that have single prover interactive protocols). Then we will discuss the [3] paper of interactive proof systems with bounded prover complexity and also include comparisons with helping machines.

Noticing the connection to interactive proof systems, in [24] Schöning generalized the class $\mathsf{P}_{1-\mathsf{help}}$ by allowing randomized robust computations. We recall the formal definition.

Definition 9. [24] *A languages L is in $\mathsf{BPP}_{1-\mathsf{help}}$ if there is a randomized polynomial-time Turing machine M and an oracle A such that for all inputs $x \in \Sigma^*$:*

- *If $x \in L$ then*
$$\text{Prob}[M^A(x)\ accepts] \geq 3/4.$$
- *If $x \notin L$ then for all oracles B*
$$\text{Prob}[M^B(x)\ accepts] \leq 1/4.$$

We now recall a quick definition of interactive proof systems. The reader can find more details in complexity theory textbooks such as [1].

Definition 10. *Let V be a probabilistic polynomial-time machine and p a polynomial. A language L is in the class MIP if there is* multiprover interactive protocol *such that for every positive integer n and $x \in \Sigma^n$:*

- *If $x \in L$ then there exist provers $P_1, P_2, \ldots, P_{p(n)}$ such that*
$$\text{Prob}[P_1, \ldots, P_{p(n)}\ make\ V\ accept] \geq 3/4.$$

- *If $x \notin L$ then for all provers $P'_1, P'_2, \ldots, P'_{p(n)}$ such that*

$$\text{Prob}[P'_1, \ldots, P'_{p(n)} \textit{ make } V \textit{ accept}] \leq 1/4.$$

We note that IP is the subclass of MIP when the interactive protocol allows only a single prover. Two seminal results in the area of interactive proof systems are: IP = PSPACE [25, 20] and MIP = NEXP [5].

It turns out that the class $\mathsf{BPP}_{1-\mathsf{help}}$ coincides with MIP, which is the class of languages that have multiprover interactive proof systems. This was actually discovered in [5] as an "oracle" characterization of MIP in the course of the proof that MIP = NEXP [5].

Theorem 11. [5] $\mathsf{BPP}_{1-\mathsf{help}} = \mathsf{MIP}$.

An interesting point that arises in interactive protocols is the complexity of the "honest provers" $P_i, 1 \leq i \leq p(n)$ for a given language L. Let us denote by $\mathsf{IP}[C]$ and $\mathsf{MIP}[C]$, the subclasses of IP and MIP, respectively, consisting of languages with honest provers that are polynomial-time Turing reducible to some set in C. With this notation we recall some known results here:

- PSPACE = IP[PSPACE] [11, 25].
- $\mathsf{P}^{\mathsf{PP}} \subseteq \mathsf{IP}[\mathsf{PP}]$ [20].
- NEXP = $\mathsf{MIP}[\mathsf{EXP}^{\mathsf{NP}}]$ [5].
- $\oplus\mathsf{P}$ and PH are contained in $\mathsf{IP}[\oplus\mathsf{P}]$ [4].

Regarding the fourth containment above more can be said. In fact, since $\mathsf{BPP}_{1-\mathsf{help}}[\oplus\mathsf{P}] = \mathsf{MIP}[\oplus\mathsf{P}] = \mathsf{IP}[\oplus\mathsf{P}]$, it follows that $\mathsf{BPP}_{1-\mathsf{help}}[\oplus\mathsf{P}] = \mathsf{BPP}^{\oplus\mathsf{P}} = \mathsf{BP}\cdot\oplus\mathsf{P}$. It is interesting to compare this with Theorem 7, which is about deterministic robust machines helped by $\oplus$P oracles.

3.1 *Interactive proof systems with provers in* P/poly

The class MIP[P/poly] is an interesting uniform subclass of P/poly. It was investigated in [3], and we recall observations from that paper in this subsection.

Proposition 12. $\mathsf{BPP} \subseteq \mathsf{MIP}[\mathsf{P/poly}] \subset \mathsf{P/poly}$.

As noted in [14, 24], for any class of languages C we have $\mathsf{BPP}_{1-\mathsf{help}}[C] = \mathsf{MIP}[C]$. In particular, we have:

Proposition 13. $\mathsf{BPP}_{1-\mathsf{help}}[\mathsf{P/poly}] = \mathsf{MIP}[\mathsf{P/poly}]$.

It follows as a consequence that $\mathsf{P}_{1-\mathsf{help}}[\mathsf{P/poly}]$ is contained in $\mathsf{MIP}[\mathsf{P/poly}]$. One might wonder if $\mathsf{MIP}[\mathsf{P/poly}]$ could contain more languages than BPP. However, it turns out that $\mathsf{MIP}[\mathsf{P/poly}]$ contains all sparse sets in NP.

Theorem 14. *All sparse sets in* NP *are contained in* $\mathsf{P}_{1-\mathsf{help}}[\mathsf{P/poly}]$.

Proof Sketch. et $S \in \mathsf{NP}$ be a sparse set. Suppose N is an NP machine accepting S. Let $S^{=n}$ denote the strings of length n in S. For each $s \in S^{=n}$, let w_s denote the leftmost accepting path in $N(s)$. We can ensure by padding that $|w_x| = p(|x|)$ for each input $x \in S$, for a fixed polynomial p. We include the string $\langle s, y, 0^n \rangle$ in the helping oracle $\hat{A}$ for every prefix y of the string w_s, for each $s \in S^{=n}$ and each n. As S is sparse, it follows that $\hat{A} \in \mathsf{P/poly}$.

Now, let x be an input instance for S. We construct a deterministic robust Turing machine M that queries a given oracle A, for strings of the kind $\langle x, y, 0^{|x|} \rangle$, to guide a prefix search for the leftmost accepting path w_x of $N(x)$. Once w_x is constructed, M accepts iff $N(x)$ accepts along w_x. Clearly, $\hat{A}$ is a helping oracle for M which proves that S is in $\mathsf{P}_{1-\mathsf{help}}[\mathsf{P/poly}]$. □

Now, since $\mathsf{P}_{1-\mathsf{help}}[\mathsf{P/poly}] \subseteq \mathsf{MIP}[\mathsf{P/poly}]$, the existence of sparse NP sets not in BPP would imply that BPP is a proper subclass of $\mathsf{MIP}[\mathsf{P/poly}]$. Applying a standard translation technique [9] we can obtain the following.

Theorem 15. *If* NE *is* not *contained in* BPE *then* BPP *is a proper subset of* $\mathsf{MIP}[\mathsf{P/poly}]$.

On the other hand, we stress that $\mathsf{MIP}[\mathsf{P/poly}]$ is a small complexity class. It is shown in [3] that, like BPP, $\mathsf{MIP}[\mathsf{P/poly}]$ is also *low* for Σ_2^p: i.e. sets in $\mathsf{MIP}[\mathsf{P/poly}]$ are powerless as oracle to Σ_2^p. More precisely,

Theorem 16. $\Sigma_2^A = \Sigma_2^p$ *for all* $A \in \mathsf{MIP}[\mathsf{P/poly}]$.

3.2 *Oblivious NP*

In [15], in their study of fixed circuit lower bounds for different complexity classes, the authors considered the notion of oblivious classes and specifically considered oblivious versions of NP and MA.

Definition 17. [15] *A language L is in oblivious* NP *if there is a polynomial-time computable binary relation R, a polynomial $p(n)$, and for all n there is a witness $w_n \in \Sigma^{p(n)}$ such that $x \in \Sigma^n$: $x \in L$ if and only if $R(x, w_n) = 1$.*

The class oblivious MA is similarly defined when the binary relation R is relaxed to be computable in randomized polynomial time with one-sided error.

Theorem 18. *Oblivious* NP *coincides with* $\mathsf{P}_{1-\mathsf{help}}[\mathsf{P/poly}]$.

Proof Sketch. uppose L is in $\mathsf{P}_{1-\mathsf{help}}[\mathsf{P}/\mathsf{poly}]$ witnessed by a polynomial-time bounded robust machine M and helping oracle $A \in \mathsf{P}/\mathsf{poly}$. For inputs of length n the robust machine M makes queries to the oracle of length bounded by $p(n)$ for a fixed polynomial p. Let w_n denote the concatenation of all advice strings upto length $p(n)$. Then we can define the required binary relation R from the robust machine M such that $x \in L \cap \Sigma^n$ if and only if $R(x, w_n) = 1$.

Conversely, suppose L is in oblivious NP. We define the "helping" oracle $\hat{A}$ consisting of all pairs $\langle 0^n, i, b\rangle$ where b is the i^{th} bit of w_n for $1 \leq i \leq p(n)$. The corresponding robust machine M on input $x \in \Sigma^n$ will query the oracle A to extract all the bits of the string w_n and then compute $R(x, w_n)$ to decide if $x \in L$. Clearly, M is robust with $\hat{A}$ as one-sided helping oracle for accepting L. □

Likewise, we can easily show that oblivious MA coincides with $\mathsf{BPP}_{1-\mathsf{help}}[\mathsf{P}/\mathsf{poly}]$.

4 Self-helpers

We now consider the notion of *self-helping* defined by Ko [18] and discuss its connection to *program checkers*. Program result checking is another idea emanating from work on interactive proofs.

Definition 19. [18] *A language $L \subseteq \Sigma^*$ is a self-helper if L is in $\mathsf{P}_{\mathsf{help}}[L]$.*

Integer factorization provides a nice example in this context. For every positive integer n we are interested in computing the function $\mathrm{fact}(n) = \langle (p_1, e_1), (p_2, e_2), \ldots, (p_k, e_k)\rangle$, such that $p_1 < p_2 \cdots < p_k$ are distinct primes and $n = \prod_i p_i^{e_i}$. Furthermore, we can assume some standard binary encoding of $\mathrm{fact}(n)$ such that $|\mathrm{fact}(n)| = c.\lceil \log n\rceil$ for a fixed constant c. We associate with $\mathrm{fact}(n)$ the language:

$$L_{\mathrm{fact}} = \{\langle n, b, i\rangle \mid 1 \leq i \leq c.\lceil \log n\rceil \text{ and the } i^{th} \text{ bit of } \mathrm{fact}(n) \text{ is } b\}.$$

Clearly, by construction, integer factorization and L_{fact} are polynomial-time equivalent.

Lemma 20. *The language L_{fact} is a self-helper.*

Proof Sketch. he polynomial-time robust machine M on input $\langle n, \alpha, i\rangle$ will query the given oracle, say A, for each $\langle n, b, j\rangle$, where $b \in \{0, 1\}$ and $1 \leq j \leq c.\lceil \log n\rceil$. For any i, if both $\langle n, 0, j\rangle$ and $\langle n, 1, j\rangle$ are in A or both are not in A then the oracle A is bad and the machine M will output ? and stop. Otherwise, from the answers to all queries the machine M can construct $\mathrm{fact}(n) = \langle (p_1, e_1), (p_2, e_2), \ldots, (p_k, e_k)\rangle$ and verify that $n = \prod_i p_i^{e_i}$ and $p_1 < p_2 < \cdots < p_k$. If the verification fails the machine will output ? and stop. If it succeeds the machine M will accept $\langle n, \alpha, i\rangle$ iff it agrees with the

oracle A's answer. Clearly the machine M is robust and accepts L_{fact}. Furthermore, the language L_{fact} clearly helps M. □

The notion of self-helping is connected to an interesting philosophical question: can there be a *nonconstructive* proof of $\mathsf{P} = \mathsf{NP}$? In other words, is it possible that we obtain a "mathematical proof" that SAT is polynomial-time solvable, but we do not have the actual algorithm? The first place where this question was studied is attributed to Levin [19] (in [13, 12]). The following definition is helpful.

Definition 21. [13] *By* P *constructively equal to* NP *is meant that an algorithm is known that computes, from the index and time-bound of an* NP *machine M, the index of a deterministic polynomial-time Turing machine for* $L(M)$*. Furthermore, a language A is* constructively NP*-hard if a polynomial-time many-one reduction from* SAT *to A is known.*

More generally, for a computational problem X (which is not known to be polynomial-time computable nor is any superpolynomial time lower bounds known for it) can we prove a theorem of the kind "If X is polynomial-time computable then it is *constructively* polynomial-time computable"?

This is made precise in [12] as an algorithm design problem: A computational problem X is P*-time self-witnessing* if we can design an algorithm $\mathcal{A}$ for X with the property that if there exists some polynomial-time algorithm $\mathcal{B}$ for X then $\mathcal{A}$ is a polynomial-time algorithm for X.

Levin [19] first noticed that integer factorization has such an algorithm. Indeed, it is a simple consequence of the fact that L_{fact} is a self-helper. In general, we have the following easy to prove observation.

Theorem 22. [2] *If we have designed a robust oracle machine M for a set A such that A is a self-helper w.r.t. M then A has the* P*-time self-witnessing property defined above.*

It is an interesting open problem if an NP-complete set has the P-time self-witnessing property. One way to prove this would be by showing that NP-complete sets are self-helpers, but that would imply $\mathsf{NP} = \mathsf{coNP}$ because self-helpers are in $\mathsf{NP} \cap \mathsf{coNP}$. However, it is easy to show that every NP-complete set A is a "one-sided" self-helper: i.e. $A \in \mathsf{P}_{1-\mathsf{help}}[A]$ for NP-complete sets A [18].

4.1 *Graph Minor Theorem*

The Graph Minor Theorem due to Robertson and Seymour [22] is a celebrated monumental result in graph theory. We recall the statement and discuss how it is connected to self-helping oracles.

A graph H is a *minor* of a graph G (denoted $H \leq_m G$), if is H can be obtained from G by a sequence of zero or more edge-contractions, edge-deletions, or vertex-deletions of G. Clearly, $\leq_m$ forms a preorder on graphs (it is not a partial order because antisymmetry is only upto isomorphism).

A class $\mathcal{K}$ of graphs is *minor-closed* if for every $G \in \mathcal{K}$ every minor of G also belongs to $\mathcal{K}$. For instance, given a family of graphs $\{G_i\}_{i\geq 0}$ we can define $\mathcal{K}$ to consist of all graphs G such that $G_i \not\leq_m G$ for all $i \geq 0$. Clearly, $\mathcal{K}$ is a minor-closed family of graphs and is defined by the list G_i of *excluded minors*.

Conversely, if $\mathcal{K}$ is any minor-closed family of graphs then it can be characterized by a list of excluded minors $\{G_i\}_{i\geq 0}$: we can simply let $G_i, i \geq 0$ be the set of all graphs not in $\mathcal{K}$.

The Graph Minor Theorem asserts that every minor-closed family of graphs can be characterized by a *finite* list of excluded minors.

Theorem 23 (Graph Minor Theorem)**.** [22] *Every minor-closed family of graphs can be characterized by a finite list of excluded minors.*

In the course of their proof (spread over 500 pages and twenty articles), Robertson and Seymour also gave an $O(n^3)$ time algorithm for checking if H is a minor of G, for every *fixed* graph H. Here n is the number of vertices in G, and the big-Oh hides a superpolynomial constant depending on H.

Consequently, for any minor-closed family $\mathcal{K}$ of graphs, there is an $O(n^3)$ algorithm to check if $G \in \mathcal{K}$. We only need to check for each excluded minor H that $H \not\leq_m G$. A curious aspect of this algorithm is that the list of excluded minors for $\mathcal{K}$ may be unknown! Moreover, the list, though finite, might be astronomically large!

This question was studied in [13] who show how to deal with this problem in many cases. Specifically, we explain an algorithm adapted from [13], in the context of self-helping oracles, that works correctly for certain minor-closed families $\mathcal{K}$, *without* knowing explicitly the list of excluded minors.

Let $G_1 \prec G_2 \prec \ldots$ be a total ordering on all undirected graphs such that $G_i \prec G_j$ iff either (i) $|V(G_i)| < |V(G_j)|$, or (ii) $|V(G_i)| = |V(G_j)|$ and $|E(G_i)| < |E(G_j)|$, or (iii) $|V(G_i)| = |V(G_j)|$ and $|E(G_i)| = |E(G_j)|$ and G_i lexicographically precedes G_j.

Theorem 24. [13] *Suppose $\mathcal{K}$ is a minor-closed family of graphs such that*

- $\mathcal{K} \in \mathsf{P}_{\mathsf{help}}[\mathcal{K}]$ *witnessed by a robust oracle machine M.*
- *The robust machine M has the additional property that for any input graph G the machine M queries the oracle only for graphs G' such that $G' \prec G$.*

Then, from M we can design a deterministic polynomial-time decision procedure for $\mathcal{K}$ (which does not require finding the entire list of excluded minors for $\mathcal{K}$).

Proof Sketch. We can assume M is polynomial-time bounded and outputs one of $Acc, Rej, ?$. We describe the polynomial-time decision procedure, call it $\mathcal{A}$, for the minor-closed family $\mathcal{K}$. Let G be an input instance whose membership in $\mathcal{K}$ we need to decide. As $\mathcal{K}$ is a minor-closed family, by the Graph Minor Theorem it is characterized by a finite (but unknown) list of excluded minors. In the course of its execution the decision procedure will discover a subset E of these excluded minors. To begin

with we can assume $E = \emptyset$. If at any stage $H \leq_m G$ for some $H \in E$ then we can reject G and stop (because H is an excluded minor).

If $H \not\leq_m G$ for any H in the current set E, the decision procedure simulates the robust machine M on input G answering the oracle queries with oracle $\mathcal{K}_E$ defined as follows:

$$\mathcal{K}_E = \{G' \mid H \not\prec G' \text{ for all } H \in E\}.$$

Notice that $\mathcal{K}_E \supset \mathcal{K}$. Thus, $G' \notin \mathcal{K}_E$ implies $G' \notin \mathcal{K}$ but not, in general, the other direction. In the end, if M accepts the input G then the decision procedure $\mathcal{A}$ also accepts (because M is robust and $\mathcal{K}$ is a self-helper which means the "yes" outputs are correct). Now, suppose M with oracle $\mathcal{K}_E$ outputs "no". If all the queries are correctly answered by oracle $\mathcal{K}_E$ then the "no" answer is correct because the machine is robust. That means E is not the complete list of excluded minors. On the other hand, if some "yes" answers by oracle $\mathcal{K}_E$ are incorrect that also implies E is not the complete list of excluded minors. In either case, the algorithm $\mathcal{A}$ will run an enumeration of all graphs in $\prec$ order and look for the first graph G' such that $G' \notin \mathcal{K}$. This is the first graph $G' \notin E$ in the $\prec$ order rejected by the machine M with $\emptyset$ as oracle. By the Graph Minor Theorem we know that E is a fixed set of constant size. If the maximum number of vertices in a graph in E is c then the above enumeration takes time $2^{c^2}\text{poly}(n)$. Hence the overall running time is polynomial in n.

□

4.2 *Program Checkers*

We briefly mention connections to the notion of program result checking defined by Blum and Kannan [8]. We first recall the formal definition.

Definition 25. [8] *A computational problem π is said to be* checkable *if there is a randomized polynomial-time oracle Turing machine C_π such that for any deterministic program P that halts on all inputs and purports to solve π, and input instance x of π and security parameter 1^k the following hold*

- *If $P(y) = \pi(y)$ for all inputs y then $C_\pi(x, 1^k) = 1$ with probability* 1.
- *If $P(x) \neq \pi(x)$ then with probability at least $1 - 1/2^k$ $C_\pi(x, 1^k)$ outputs P is incorrect.*

The problem π is deterministically checkable *if C_π is a deterministic oracle Turing machine.*

A host of natural computational problems is shown to have efficient program checkers in [8] and many subsequent papers. Standard complexity-theoretic examples include the Permanent, complete problems for $\oplus$P, PSPACE, and EXP.

It is easy to prove that integer factorization has a deterministic checker using the fact that L_{fact} is a self-helper. Indeed, it can be shown that a decision problem π is deterministically checkable iff it is a self-helper.

References

[1] S. Arora, B. Barak. *Computational Complexity: A modern approach.* Cambridge University Press, New York, USA, 2009.

[2] V. Arvind. Constructivizing membership proofs in complexity classes. *Intl Journal of Foundations of Computer Science,* 8(4):433-442, 1997.

[3] V. Arvind, J. Köbler, R. Schuler. On helping and interactive proof systems. *Intl. Journal of Foundations of Computer Science,* 6(2): 137-153, 1995.

[4] L. Babai, L. Fortnow. Arithmetization: a new method in structural complexity. *Computational Complexity,* 1:41-66, 1991.

[5] L. Babai, L. Fortnow, C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity,* 1:1-40, 1991.

[6] J. Balcázar. Only smart oracles help. Technical report LSI-88-9, Universitat Politècnica de Catalunya, 1988.

[7] J. Balcázar. Self-reducibility structures and solutions of NP problems. *Revista Matematica de la Universidad Complutense de Madrid,* 2:175-184, 1989.

[8] M. Blum, S. Kannan. Designing programs that check their work. *Journal of the ACM,* 42(1):269-291, 1995.

[9] R. Book. Tally languages and complexity classes. *Information and Control,* 26:186-193, 1974.

[10] J.Y. Cai, L. Hemachandra, J. Vyskoc. Promises and fault-tolerant database access, In *Complexity Theory,* volume edited by K. Ambos-Spies, S. Homer, U. Schöning, pages 101-146, CUP, 1993.

[11] P. Feldman. The optimal prover lives in PSPACE. manuscript, 1986.

[12] M.R. Fellows, N. Koblitz. Self-witnessing polynomial time complexity and prime factorization. *Proc. 6th Annual IEEE Structure in Complexity Conference,* 107-110, 1992.

[13] M.R. Fellows, M.A. Langston. Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM,* 35(3):727-739, 1988.

[14] L. Fortnow, J. Rompel, M. Sipser. On the power of multiprover interactive protocols. *Proc. 3rd Structure in Complexity Theory Conference,* 156-161, 1988.

[15] L. Fortnow, R. Santhanam, R. Williams. Fixed-Polynomial Size Circuit Bounds. *24th Annual IEEE Conference on Computational Complexity,* 19-26, 2009.

[16] S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing,* 18:186-208, 1989.

[17] J. Hartmanis, L. Hemachandra. Robust machines accept easy sets. *Theoretical Computer Science,* 74(2):217-226, 1984.

[18] Ker-I Ko. On helping by robust oracle machines. *Theoretical Computer Science,* 52:15-36, 1987.

[19] L.A. Levin. Universal Enumeration problems. *Problemy Peredachi Informatsii,* IX, 115-116, 1972.

[20] C. Lund, L. Fortnow, H. Karloff, N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM,* 39(4):859-868, 1992.

[21] M. Ogihara. On helping by parity-like languages. *Theoretical Computer Science,* 54:41-43, 1995.

[22] N. Robertson, P. Seymour. Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory,* Series B, 92(2):325ÎšâĆňâĂIJ357, 2004.

[23] U. Schöning. Robust algorithms: a different approach to oracles. *Theoretical Computer Science,* 40:57-66, 1985.

[24] U. Schöning. Robust oracle machines. *Proc. 13th Mathematical Foundations of Computer Science,* LNCS, Springer, 93-106, 1988.

[25] A. Shamir. IP=PSPACE. *Journal of the ACM,* 39(4):869-877, 1992.

The Distributed Computing Column

BY

Panagiota Fatourou

Department of Computer Science, University of Crete
P.O. Box 2208 GR-714 09 Heraklion, Crete, Greece
and
Institute of Computer Science (ICS)
Foundation for Research and Technology (FORTH)
N. Plastira 100. Vassilika Vouton
GR-700 13 Heraklion, Crete, Greece
faturu@csd.uoc.gr

AND BY

Stefan Schmid

Technical University of Berlin
D-10587 Berlin, Germany
and
Telekom Innovation Laboratories (T-Labs)
Ernst Reuter Platz 7, D - 10587 Berlin, Germany
schmiste@gmail.com

After serving for more than 6 years as the editor of the Distributed Computing Column (DC) of the Bulletin of the European Association for Theoretical Computer Science (BEATCS), I am glad to leave the column in the hands of Stefan Schmid, who will be the next editor. Stefan and I will co-edit the DC column for this and the next issue of BEATCS and he will replace me right after. I wish Stefan that his tenure as editor will be as pleasurable as mine was. I am really grateful to the authors that have contributed to the column during these years and I deeply thank them for their great contributions!

Panagiota Fatourou

I am very happy to take over the responsibility for the Distributed Computing column of the BEATCS. My goal is to continue providing an interesting column format where researchers have the opportunity to give survey-like but technical overviews of recent advances in distributed computing, written for a broad TCS audience. I would like to thank Panagiota for her great work and commitment over the last years, and I am very proud to be able to co-edit two issues together with her. I wish Panagiota all the best for the future. I would also like to thank Jukka Suomela for his excellent survey on recent advances in lower bounds for distributed graph algorithms.

Stefan Schmid

A Closer Look at Concurrent Data Structures and Algorithms

Gadi Taubenfeld
The Interdisciplinary Center, P.O.Box 167, Herzliya 46150 Israel
tgadi@idc.ac.il

Abstract

In this survey article, I will present three ideas, regarding the construction of concurrent data structures and algorithms, recently published in [27, 28, 29].

The first idea is that of *contention-sensitivity*. A contention-sensitive data structure is a concurrent data structure in which the overhead introduced by locking is eliminated in common cases, when there is no contention, or when processes with non-interfering operations access it concurrently. This notion is formally defined, several contention-sensitive data structures are presented, and transformations that facilitate devising such data structures are discussed.

The second idea is the introduction of a new synchronization problem, called *fair synchronization*. Solving the new problem enables to automatically add strong fairness guarantees to existing implementations of concurrent data structures, without using locks, and to transform any solution to the mutual exclusion problem into a fair solution.

The third idea is a generalization of the traditional notion of *fault tolerance*. Important classical problems have no asynchronous solutions which can tolerate even a single fault. It is shown that while some of these problems have solutions which guarantee that in the presence of *any* number of faults *most* of the correct processes will terminate, other problems do not even have solutions which guarantee that in the presence of just *one* fault at least one correct process terminates.

1 Introduction

Concurrent access to a data structure shared among several processes must be synchronized in order to avoid interference between conflicting operations. Mutual exclusion locks are the de facto mechanism for concurrency control on concurrent data structures: a process accesses the data structure only inside a critical section

code, within which the process is guaranteed exclusive access. Any sequential data structure can be easily made concurrent using such a locking approach. The popularity of this approach is largely due to the apparently simple programming model of such locks, and the availability of lock implementations which are reasonably efficient.

Using locks may, in various scenarios, degrade the performance of concurrent applications, as it enforces processes to wait for a lock to be released. Moreover, slow or stopped processes may prevent other processes from ever accessing the data structure. Locks can introduce false conflicts, as different processes with non-interfering operations contend for the same lock, only to end up accessing disjoint data. This motivates attempts to design data structures that avoid locking.

The advantages of concurrent data structures and algorithms which completely avoid locking are that they are not subject to priority inversion, they are resilient to failures, and they do not suffer significant performance degradation from scheduling preemption, page faults or cache misses. On the other hand, such algorithms may impose too much overhead upon the implementation and are often complex and memory consuming.

We consider an intermediate approach for the design of concurrent data structures. A *contention-sensitive* data structure is a concurrent data structure in which the overhead introduced by locking is eliminated in common cases, when there is no contention, or when processes with non-interfering operations access it concurrently. When a process invokes an operation on a contention-sensitive data structure, in the absence of contention or interference, the process must be able to complete its operation in a small number of steps and without using locks. Using locks is permitted only when there is interference [27]. In Section 2, the notion of contention-sensitivity is formally defined, several contention-sensitive data structures are presented, and transformations that facilitate devising such data structures are discussed.

Most published concurrent data structures which completely avoid locking do not provide any fairness guarantees. That is, they allow processes to access a data structure and complete their operations arbitrarily many times before some other trying process can complete a single operation. In Section 3, we show how to automatically transfer a given concurrent data structure which avoids locking and waiting into a similar data structure which satisfies a strong fairness requirement, without using locks and with limited waiting. To achieve this goal, we introduce and solve a *new* synchronization problem, called *fair synchronization* [29]. Solving the new problem enables us to add fairness to existing implementations of concurrent data structures, and to transform any lock (i.e., a solution to the mutual exclusion problem) into a fair lock.

In Section 4, we generalize the traditional notion of fault tolerance in a way which enables to capture more sensitive information about the resiliency of a con-

current algorithm. Intuitively, an algorithm that, in the presence of any number of faults, always guarantees that all the correct processes, except maybe one, successfully terminate (their operations), is more resilient to faults than an algorithm that in the presence of a single fault does not even guarantee that a single correct process ever terminates. However, according to the standard notion of fault tolerance both algorithms are classified as algorithms that can not tolerate a single fault. Our general definition distinguishes between these two cases.

It is well known that, in an asynchronous system where processes communicate by reading and writing atomic read/write registers important classical problems such as, consensus, set-consensus, election, perfect renaming, implementations of a test-and-set bit, a shared stack, a swap object and a fetch-and-add object, have no deterministic solutions which can tolerate even a single fault. In Section 4, we show that while, some of these classical problems have solutions which guarantee that in the presence of *any* number of faults most of the correct processes will successfully terminate; other problems do not even have solutions which guarantee that in the presence of just *one* fault at least one correct process terminates.

Our model of computation consists of an asynchronous collection of n processes which communicate asynchronously via shared objects. Asynchrony means that there is no assumption on the relative speeds of the processes. Processes may fail by crashing, which means that a failed process stops taking steps forever. In most cases, we assume that processes communicate by reading and writing atomic registers. By atomic registers we always mean atomic read/write registers. In few cases, we will also consider stronger synchronization primitives. With an atomic register, it is assumed that operations on the register occur in some definite order. That is, reading or writing an atomic register is an indivisible action.

In a model where participation is required, every correct process must eventually execute its code. A more interesting and practical situation is one in which participation is *not* required, as assumed when solving resource allocation problems or when designing concurrent data structures. In this paper we always assume that participation is not required.

2 Contention-sensitive Data Structures

2.1 Motivation

As already mentioned in Section 1, using locks may, in various scenarios, degrade the performance of concurrent data structures. On the other hand, concurrent data structures which completely avoid locking may impose too much overhead upon the implementation and are often complex and memory consuming.

We propose an intermediate approach for the design of concurrent data structures. While the approach guarantees the correctness and fairness of a concurrent data structure under all possible scenarios, it is especially efficient in common cases when there is no (or low) contention, or when processes with non-interfering operations access a data structure concurrently.

2.2 Contention-sensitive data structures: The basic idea

Contention for accessing a shared object is usually rare in well designed systems. Contention occurs when multiple processes try to acquire a lock at the same time. Hence, a desired property in a lock implementation is that, in the absence of contention, a process can acquire the lock extremely fast, without unnecessary delays. Furthermore, such fast implementations decrease the possibility that processes which invoke operations on the same data structure in about the same time but not simultaneously, will interfere with each other. However, locks were introduced in the first place to resolve conflicts when there is contention, and acquiring a lock *always* introduces some overhead, even in the cases where there is no contention or interference.

We propose an approach which, in common cases, eliminates the overhead involved in acquiring a lock. The idea is simple: assume that, for a given data structure, it is known that in the absence of contention or interference it takes some fixed number of steps, say at most 10 steps, to complete an operation, not counting the steps involved in acquiring and releasing the lock. According to our approach, when a process invokes an operation on a given data structure, it first tries to complete its operation, by executing a short code, called the *shortcut code*, which does not involve locking. Only if it does not manage to complete the operation fast enough, i.e., within 10 steps, it tries to access the data structure via locking. The shortcut code is required to be *wait-free*. That is, its execution by a process takes only a finite number of steps and always terminates, regardless of the behavior of the other processes.

Using an efficient shortcut code, although eliminates the overhead introduced by locking in common cases, introduces a major problem: we can no longer use a sequential data structure as the basic building block, as done when using the traditional locking approach. The reason is simple, many processes may access the same data structure simultaneously by executing the shortcut code. Furthermore, even when a process acquires the lock, it is no longer guaranteed to have exclusive access, as another process may access the same data structure simultaneously by executing the shortcut code.

Thus, a central question which we are facing is: if a sequential data structure can not be used as the basic building block for a general technique for constructing a contention-sensitive data structure, then what is the best data structure to use?

Before we proceed to discuss formal definitions and general techniques, which will also help us answering the above question, we demonstrate the idea of using a shortcut code (which does not involve locking), by presenting a contention-sensitive solution to the binary consensus problem using atomic read/write registers and a single lock.

2.3 A simple example: Contention-sensitive consensus

The *consensus problem* is to design an algorithm in which all correct processes reach a common decision based on their initial opinions. A consensus algorithm is an algorithm that produces such an agreement. While various decision rules can be considered such as "majority consensus", the problem is interesting even when the decision value is constrained only when all processes are unanimous in their opinions, in which case the decision value must be the common opinion. A consensus algorithm is called *binary* consensus when the number of possible initial opinions is two.

Processes are not required to participate in the algorithm. However, once a process starts participating it is guaranteed that it may fail only while executing the shortcut code. We assume that processes communicate via atomic registers. The algorithm uses an array $x[0..1]$ of two atomic bits, and two atomic registers y and *out*. After a process executes a **decide**() statement, it immediately terminates.

CONTENTION-SENSITIVE BINARY CONSENSUS: program for process p_i with input $in_i \in \{0, 1\}$.

shared $x[0..1]$: array of two atomic bits, initially both 0
y, *out* : atomic registers which range over $\{\perp, 0, 1\}$, initially both $\perp$

1 $x[in_i] := 1$ // start shortcut code
2 **if** $y = \perp$ **then** $y := in_i$ **fi**
3 **if** $x[1 - in_i] = 0$ **then** $out := in_i$; **decide**(in_i) **fi**
4 **if** $out \neq \perp$ **then decide**(out) **fi** // end shortcut code
5 lock **if** $out = \perp$ **then** $out := y$ **fi** unlock ; **decide**(out) // locking

When a process runs alone (either before or after a decision is made), it reaches a decision after accessing the shared memory at most five times. Furthermore, when all the concurrently participating processes have the same preference – i.e., when there is no interference – a decision is also reached within five steps and without locking. Two processes with conflicting preferences, which run at the same time, will not resolve the conflict in the shortcut code if both of them find $y = \perp$. In such a case, some process acquires the lock and sets the value of *out* to be the final decision value. The assignment $out := y$ requires two memory references and hence it involves two atomic steps.

2.4 Progress conditions

Numerous implementations of locks have been proposed over the years to help coordinating the activities of the various processes. We are not interested in implementing new locks, but rather assume that we can use existing locks. We are not at all interested whether the locks are implemented using atomic registers, semaphores, etc. We do assume that a lock implementation guarantees that: (1) no two processes can acquire the same lock at the same time, (2) if a process is trying to acquire the lock, then in the absence of failures some process, not necessarily the same one, eventually acquires that lock, and (3) the operation of releasing a lock is wait-free.

Several progress conditions have been proposed for data structures which avoid locking, and in which processes may fail by crashing. *Wait-freedom* guarantees that *every* active process will always be able to complete its pending operations in a finite number of steps [8]. *Non-blocking* (which is sometimes called lock-freedom) guarantees that *some* active process will always be able to complete its pending operations in a finite number of steps [13]. *Obstruction-freedom* guarantees that an active process will be able to complete its pending operations in a finite number of steps, if all the other processes "hold still" long enough [9]. Obstruction-freedom does not guarantee progress under contention.

Several progress conditions have been proposed for data structures which may involve waiting. *Livelock-freedom* guarantees that processes not execute forever without making forward progress. More formally, livelock-freedom guarantees that, in the absence of process failures, if a process is active, then *some* process, must eventually complete its operation. A stronger property is *starvation-freedom* which guarantees that each process will eventually make progress. More formally, starvation-freedom guarantees that, in the absence of process failures, every active process must eventually complete its operation.

2.5 Defining contention-sensitive data structures

An implementation of a contention-sensitive data structure is divided into *two* continuous sections of code: the *shortcut code* and the *body code*. When a process invokes an operation it first executes the shortcut code, and if it succeeds to complete the operation, it returns. Otherwise, the process tries to complete its operation by executing the body code, where it usually first tries to acquire a lock. If it succeeds to complete the operation, it releases the acquired lock(s) and returns. The problem of implementing a contention-sensitive data structure is to write the *shortcut code* and the *body code* in such a way that the following *four* requirements are satisfied,

- **Fast path:** In the absence of contention or interference, each operation must

be completed while executing the shortcut code only.

- **Wait-free shortcut:** The shortcut code must be wait-free – its execution should require only a finite number of steps and must always terminate. (Completing the shortcut code does not imply completing the operation.)
- **Livelock-freedom:** In the absence of process failures, if a process is executing the shortcut code or the body code, then some process, not necessarily the same one, must eventually complete its operation.
- **Linearizability:** Although operations of concurrent processes may overlap, each operation should appear to take effect instantaneously. In particular, operations that do not overlap should take effect in their "real-time" order.

In [27], several additional desirable properties are defined.

2.6 An example: Contention-sensitive election

The *election problem* is to design an algorithm in which all participating processes choose one process as their leader. More formally, each process that starts participating eventually decides on a value from the set $\{0, 1\}$ and terminates. It is required that exactly one of the participating processes decides 1. The process that decides 1 is the elected leader. Processes are not required to participate. However, once a process starts participating it is guaranteed that it will not fail. It is known that in the presence of one crash failure, it is not possible to solve election using only atomic read/write registers [18, 31].

The following algorithm solves the election problem for any number of processes, and is related to the splitter constructs from [14, 16, 17]. A single lock is used. It is assumed that after a process executes a **decide**() statement, it immediately terminates.

CONTENTION-SENSITIVE ELECTION: Process i's program

shared x, z: atomic registers, initially $z = 0$ and the initial value of x is immaterial
b, y, *done*: atomic bits, initially all 0
local *leader*: local register, the initial value is immaterial

```
1   x := i                                              // begin shortcut
2   if y = 1 then b := 1; decide(0) fi                  // I am not the leader
3   y := 1
4   if x = i then z := i; if b = 0 then decide(1) fi fi // I am the leader!
                                                        // end shortcut
```

```
5    lock                                                          // locking
6    if z = i ∧ done = 0 then leader = 1                           // I am the leader!
7         else await b ≠ 0 ∨ z ≠ 0
8              if z = 0 ∧ done = 0 then leader = 1; done := 1      // I am the leader!
9                   else leader = 0                                // I am not the leader
10             fi
11   fi
12   unlock ; decide(leader)                                       // unlocking
```

When a process runs alone before a leader is elected, it is elected and terminates after accessing the shared memory *six* times. Furthermore, all the processes that start running *after* a leader is elected terminate after three steps. The algorithm does not satisfy the disable-free shortcut property: a process that fails just before the assignment to b in line 2 or fails just before the assignment to z in line 4, may prevent other processes spinning in the *await* statement (line 7) from terminating.

2.7 Additional results

The following additional results are presented in [27].

- A contention-sensitive double-ended queue. To increase the level of concurrency, *two* locks are used: one for the left-side operations and the other for the right-side operations

- Transformations that facilitate devising contention-sensitive data structures. The first transformation converts any contention-sensitive data structure which satisfies livelock-freedom into a corresponding contention-sensitive data structure which satisfies starvation-freedom. The second transformation, converts any obstruction-free data structure into the corresponding contention-sensitive data structure which satisfies livelock-freedom.

- Finally, the notion of a *k-contention-sensitive* data structure in which locks are used only when contention goes above k is presented. This notion is illustrated by implementing a 2-contention-sensitive consensus algorithm. Then, for each $k \geq 1$, a progress condition, called *k-obstruction-freedom*, is defined and a transformation is presented that converts any k-obstruction-free data structure into the corresponding k-contention-sensitive data structure which satisfies livelock-freedom.

3 Fair Synchronization

3.1 Motivation

As already discussed in Section 1, using locks may degrade the performance of synchronized concurrent applications, and hence much work has been done on the design of wait-free and non-blocking data structures. However, the wait-freedom and non-blocking progress conditions do not provide fairness guarantees. That is, such data structures may allow processes to complete their operations arbitrarily many times before some other trying process can complete a single operation. Such a behavior may be prevented when fairness is required. However, fairness requires waiting or helping.

Using helping techniques (without waiting) may impose too much overhead upon the implementation, and are often complex and memory consuming. Does it mean that for enforcing fairness it is best to use locks? The answer is negative. We show how any wait-free and any non-blocking implementation can be automatically transformed into an implementation which satisfies a very strong fairness requirement without using locks and with limited waiting.

We require that no beginning process can complete two operations on a given resource while some other process is kept waiting on the same resource. Our approach, allows as many processes as possible to access a shared resource at the same time as long as fairness is preserved. To achieve this goal, we introduce and solve a new synchronization problem, called *fair synchronization*. Solving the fair synchronization problem enables us to add fairness to existing implementations of concurrent data structures, and to transform any solution to the mutual exclusion problem into a fair solution.

3.2 The fair synchronization problem

The fair synchronization problem is to design an algorithm that guarantees fair access to a shared resource among a number of participating processes. Fair access means that no process can access a resource twice while some other trying process cannot complete a single operation on that resource. There is no a priori limit (smaller than the number of processes n) on the number of processes that can access a resource simultaneously. In fact, a desired property is that as many processes as possible will be able to access a resource at the same time as long as fairness is preserved.

It is assumed that each process is executing a sequence of instructions in an infinite loop. The instructions are divided into four continuous sections: the remainder, entry, fair and exit. Furthermore, it is assumed that the entry section consists of two parts. The first part, which is called the *doorway*, is *fast wait-free*:

its execution requires only a (very small) *constant* number of steps and hence always terminates; the second part is the *waiting* part which includes (at least one) loop with one or more statements. Like in the case of the doorway, the exit section is also required to be fast wait-free. A *waiting* process is a process that has finished its doorway code and reached the waiting part of its entry section. A *beginning* process is a process that is about to start executing its entry section.

A process is *enabled* to enter its fair section at some point in time, if sufficiently many steps of that process will carry it into the fair section, independently of the actions of the other processes. That is, an enabled process does not need to wait for an action by any other process in order to complete its entry section and to enter its fair section, nor can an action by any other process prevent it from doing so.

The **fair synchronization problem** is to write the code for the entry and the exit sections in such a way that the following three basic requirements are satisfied.

- **Progress:** *In the absence of process failures and assuming that a process always leaves its fair section, if a process is trying to enter its fair section, then some process, not necessarily the same one, eventually enters its fair section.*

  The terms deadlock-freedom and livelock-freedom are used in the literature for the above progress condition, in the context of the mutual exclusion problem.

- **Fairness:** *A beginning process cannot execute its fair section twice before a waiting process completes executing its fair and exit sections once. Furthermore, no beginning process can become enabled before an already waiting process becomes enabled.*

  It is possible that a beginning process and a waiting process will become enabled at the same time. However, no beginning process can execute its fair section twice while some other process is kept waiting. The second part of the fairness requirement is called *first-in-first-enabled*. The term *first-in-first-out* (FIFO) fairness is used in the literature for a slightly stronger condition which guarantees that: no beginning process can pass an already waiting process. That is, no beginning process can enter its fair section before an already waiting process does so.

- **Concurrency:** *All the waiting processes which are not enabled become enabled at the same time.*

> It follows from the *progress* and *fairness* requirements that *all* the waiting processes which are not enabled will eventually become enabled. The concurrency requirement guarantees that becoming enabled happens simultaneously, for all the waiting processes, and thus it guarantees that many processes will be able to access their fair sections at the same time as long as fairness is preserved. We notice that no lock implementation may satisfy the concurrency requirement.

The processes that have already passed through their doorway can be divided into two groups. The enabled processes and those that are not enabled. It is not possible to always have all the processes enabled due to the fairness requirement. All the enabled processes can immediately proceed to execute their fair sections. The waiting processes which are not enabled will eventually simultaneously become enabled, before or once the currently enabled processes exit their fair and exit sections. We observe that the stronger FIFO fairness requirement, the progress requirement and concurrency requirement cannot be mutually satisfied.

3.3 The fair synchronization algorithm

We use one atomic bit, called *group*. The first thing that process i does in its entry section is to read the value of the *group* bit, and to determine to which of the two groups (0 or 1) it should belong. This is done by setting i's single-writer register $state_i$ to the value read.

Once i chooses a group, it waits until its group has priority over the other group and then it enters its fair section. The order in which processes can enter their fair sections is defined as follows: If two processes belong to different groups, the process whose group, as recorded in its *state* register, is *different* from the value of the bit *group* is enabled and can enter its fair section, and the other process has to wait. If all the active processes belong to the same group then they can all enter their fair sections.

Next, we explain when the shared *group* bit is updated. The first thing that process i does when it leaves its fair section (i.e., its first step in its exit section) is to set the *group* bit to a value which is *different* from the value of its $state_i$ register. This way, i gives priority to waiting processes which belong to the same group that it belongs to.

Until the value of the *group* bit is first changed, all the active processes belong to the same group, say group 0. The first process to finish its fair section flips the value of the *group* bit and sets it to 1. Thereafter, the value read by all the new beginning processes is 1, until the group bit is modified again. Next, *all* the processes which belong to group 0 enter and then exit their fair sections possibly at the same time until there are no active processes which belong to group 0. Then

all the processes from group 1 become enabled and are allowed to enter their fair sections, and when each one of them exits it sets to 0 the value of the *group* bit, which gives priority to the processes in group 1, and so on.

The following registers are used: (1) a single multi-writer atomic bit named *group*, (2) an array of single-writer atomic registers $state[1..n]$ which range over $\{0, 1, 2, 3\}$. To improve readability, we use below subscripts to index entries in an array. At any given time, process i can be in one of four possible states, as recorded in it single-writer register $state_i$. When $state_i = 3$, process i is not active, that is, it is in its remainder section. When $state_i = 2$, process i is active and (by reading *group*) tries to decide to which of the two groups, 0 or 1, it should belong. When $state_i = 1$, process i is active and belongs to group 1. When $state_i = 0$, process i is active and belongs to group 0.

The statement **await** *condition* is used as an abbreviation for **while** ¬*condition* **do** *skip*. The *break* statement, like in C, breaks out of the smallest enclosing *for* or *while* loop. Finally, whenever two atomic registers appear in the same statement, two separate steps are required to execute this statement. The algorithm is given below.[1]

A Fair Synchronization Algorithm: `process i's code` $(1 \le i \le n)$

Shared variables:

group: atomic bit; the initial value of the group bit is immaterial.
$state[1..n]$: array of atomic registers, which range over $\{0, 1, 2, 3\}$
Initially $\forall i : 1 \le i \le n : state_i = 3$ `/* processes are inactive */`

```
1   state_i := 2                                          /* begin doorway */
2   state_i := group                          /* choose group and end doorway */
3   for j = 1 to n do                                     /* begin waiting */
4       if (state_i ≠ group) then break fi          /* process is enabled */
5       await state_j ≠ 2
6       if state_j = 1 − state_i                      /* different groups */
7       then await   (state_j ≠ 1 − state_i) ∨ (state_i ≠ group) fi
8   od                                                    /* end waiting */
9   fair section
10  group := 1 − state_i                                   /* begin exit */
11  state_i := 3                                             /* end exit */
```

In line 1, process i indicates that it has started executing its doorway code. Then,

[1] To simplify the presentation, when the code for a fair synchronization algorithm is presented, only the entry and exit codes are described, and the remainder code and the infinite loop within which these codes reside are omitted.

in *two* atomic steps, it reads the value of *group* and assigns the value read to $state_i$ (line 2).

After passing its doorway, process i waits in the *for loop* (lines 3–8), until all the processes in the group to which it belongs are simultaneously enabled and then it enters its fair section. This happens when either, ($state_i \neq group$), i.e. the value the *group* bit points to the group which i does *not* belong to (line 4), or when all the waiting processes (including i) belong to the same group (line 7). Each one of the terms of the await statement (line 7) is evaluated separately. In case processes i and j belong to different groups (line 6), i waits until either (1) j is not competing any more or j has reentered its entry section, or (2) i has priority over j because $state_i$ is *different* than the value of the *group* bit.

In the exit code, i sets the *group* bit to a value which is different than the group to which it belongs (line 10), and changes its state to not active (line 11). We notice that the algorithm is also correct when we replace the order of lines 9 and 10, allowing process i to write the group bit immediately before it enters its fair section. The order of lines 10 and 11 is crucial for correctness.

We observe that a *non* beginning process, say p, may enter its fair section ahead of another waiting process, say q, twice: the first time if p is enabled on the other group, and the second time if p just happened to pass q which is waiting on the same group and enters its fair section first. We point out that omitting lines 1 and 5 will result in an incorrect solution. It is possible to replace each one of the 4-valued single-writer atomic registers, by three *separate* atomic bits.

An *adaptive* algorithm is an algorithm which its time complexity is a function of the actual number of participating processes rather than a function of the total number of processes. An adaptive fair synchronization algorithm using atomic register is presented in [29]. It is also shown in [29] that $n - 1$ read/write registers and conditional objects are necessary for solving the fair synchronization problem for n processes. A conditional operation is an operation that changes the value of an object only if the object has a particular value. A *conditional object* is an object that supports only conditional operations. Compare-and-swap and test-and-set are examples of conditional objects.

3.4 Fair data structures and fair locks

In order to impose fairness on a concurrent data structure, concurrent accesses to a data structure can be synchronized using a fair synchronization algorithm: a process accesses the data structure only inside a fair section. Any data structure can be easily made fair using such an approach, without using locks and with limited waiting. The formal definition of a fair data structure can be found in [29].

We name a solution to the fair synchronization problem a (finger) *ring*.[2] Using a single *ring* to enforce fairness on a concurrent data structure, is an example of coarse-grained *fair* synchronization. In contrast, fine-grained *fair* synchronization enables to protect "small pieces" of a data structure, allowing several processes with *different* operations to access it completely independently. For example, in the case of adding fairness to an existing wait-free queue, it makes sense to use two rings: one for the enqueue operations and the other for the dequeue operations.

We assume the reader is familiar with the definition of a deadlock-free mutual exclusion algorithm (DF-ME). By composing a fair synchronization algorithm (FS) and a DF-ME, it is possible to construct a *fair* mutual exclusion algorithm (FME), i.e., a fair lock. The entry section of the composed FME algorithm consists of the entry section of the FS algorithm followed by the entry section of the ME algorithm. The exit section of the FME algorithm consists of the exit section of the ME algorithm followed by the exit section of the FS algorithm. The doorway of the FME algorithm is the doorway of the FS algorithm.

4 Fault Tolerance

4.1 Motivation

According to the standard notion of fault tolerance, an algorithm is t-resilient if in the presence of up to t faults, *all* the correct processes can still complete their operations and terminate. Thus, an algorithm is *not* t-resilient, if as a result of t faults there is *some* correct process that can not terminate. This traditional notion of fault tolerance is not sensitive to the *number* of correct processes that may or may not complete their operations as a result of the failure of other processes.

Consider for example the renaming problem, which allows processes, with distinct initial names from a large name space, to get distinct new names from a small name space. A renaming algorithm that, in the presence of any number of faults, always guarantees that *most* of the correct processes, but not necessarily all, get distinct new names is clearly more resilient than a renaming algorithm that in the presence of a single fault does not guarantee that even one correct process ever gets a new name. However, using the standard notion of fault tolerance, it is not possible to compare the resiliency of such algorithms – as both are simply not even 1-resilient. This motivates us to suggest and investigate a more general notion of fault tolerance.

We generalize the traditional notion of fault tolerance by allowing a limited number participating correct processes not to terminate in the presence of faults.

[2] Many processes can simultaneously pass through the ring's hole, but the size of the ring may limit their number.

Every process that do terminate is required to return a correct result. Thus, our definition guarantees safety but may sacrifice liveness (termination), for a limited number of processes, in the presence of faults. The consequences of violating liveness are often less severe than those of violating safety. In fact, there are systems that can detect and abort processes that run for too long. Sacrificing liveness of few processes allows us to increase the resiliency of the whole system.

4.2 A general definition of fault tolerance

For the rest of the section, n denotes the number of processes, t denotes the number of faulty processes, and $N = \{0, 1, ..., n\}$.

> **Definition:** For a given function $f : N \to N$, an algorithm is (t, f)-*resilient* if in the presence of t' faults at most $f(t')$ participating correct processes may *not* terminate their operations, for every $0 \leq t' \leq t$.

It seems that (t, f)-*resiliency* is interesting only when requiring that $f(0) = 0$. That is, in the absence of faults all the participating processes must terminate their operations. The standard definition of t-resiliency is equivalent to (t, f)-resiliency where $f(t') = 0$ for every $0 \leq t' \leq t$. Thus, the familiar notion of *wait-freedom* is equivalent to $(n-1, f)$-resiliency where $f(t') = 0$ for every $0 \leq t' \leq n-1$. The new notion of (t, f)-resiliency is quite general, and in this section we focus mainly on the following three levels of resiliency.

- An algorithm is *almost-t-resilient* if it is (t, f)-*resilient*, for a function f where $f(0) = 0$ and $f(t') = 1$, for every $1 \leq t' \leq t$. Thus, in the presence of any number of up to t faults, all the correct participating processes, except maybe one process, must terminate their operations.

- An algorithm is *partially-t-resilient* if it is (t, f)-*resilient*, for a function f where $f(0) = 0$ and $f(t') = t'$, for every $1 \leq t' \leq t$. Thus, in the presence of any number $t' \leq t$ faults, all the correct participating processes, except maybe t' of them must terminate their operations.

- An algorithm is *weakly-t-resilient* if it is (t, f)-*resilient*, for a function f where $f(0) = 0$, and in the presence of any number of up to $t \geq 1$ faults, if there are *two* or more correct participating processes then one correct participating process must terminate its operation. (Notice that for $n = 2$, if one process fails the other one is not required to terminate.)

For $n \geq 3$ and $t < n/2$, the notion of weakly-t-resiliency is strictly weaker than the notion of partially-t-resiliency. For $n \geq 3$, the notion of weakly-t-resiliency is strictly weaker than the notion of almost-t-resiliency. For $n \geq 3$ and $t \geq 2$,

the notion of partially-t-resiliency is strictly weaker than the notion of almost-t-resiliency. For all n, partially-1-resiliency and almost-1-resiliency are equivalent. For $n = 2$, these three notions are equivalent. We say that an algorithm is *almost-wait-free* if it is *almost*-$(n-1)$-*resilient*, i.e., in the presence of any number of faults, all the participating correct processes, except maybe one process, must terminate. We say that an algorithm is *partially-wait-free* if it is *partially*-$(n-1)$-*resilient*, i.e., in the presence of any number of $t \leq n-1$ faults, all the correct participating processes, except maybe t of them must terminate.

4.3 Example: An almost-wait-free symmetric test-and-set bit

A test-and-set bit supports two atomic operations, called *test-and-set* and *reset*. A test-and-set operation takes as argument a shared bit b, assigns the value 1 to b, and returns the previous value of b (which can be either 0 or 1). A reset operation takes as argument a shared bit b and writes the value 0 into b.

The *sequential specification* of an object specifies how the object behaves in sequential runs, that is, in runs when its operations are applied sequentially. The sequential specification of a test-and-set bit is quite simple. In sequential runs, the first test-and-set operation returns 0, a test-and-set operation that happens immediately after a reset operation also returns 0, and all other test-and-set operations return 1. The consistency requirement is linearizability.

The algorithm below is for n processes each with a unique identifier taken from some (possibly infinite) set which does not include 0. It makes use of exactly n registers which are long enough to store a process identifier and one atomic bit. The algorithm is based on the symmetric mutual exclusion algorithm from [24].[3]

The algorithm uses a register, called *turn*, to indicate who has priority to return 1, $n-1$ *lock* registers to ensure that at most one process will return 1 between resets, and a bit, called *winner*, to indicate whether some process already returned 1. Initially the values of all these shared registers are 0. In addition, each process has a private boolean variable called *locked*. We denote by *b.turn*, *b.winner* and *b.lock*[*] the shared registers for the implementation of a specific test-and-set bit, named b.

[3] A symmetric algorithm is an algorithm in which the only way for distinguishing processes is by comparing identifiers, which are unique. Identifiers can be written, read and compared, but there is no way of looking inside any identifier. Thus, identifiers cannot be used to index shared registers.

AN ALMOST-WAIT-FREE SYMMETRIC TEST-AND-SET BIT: `process p's program.`

```
function test-and-set (b:bit) return:value in {0, 1};                    /* access bit b */
1    if b.turn ≠ 0 then return(0) fi;                                        /* lost */
2    b.turn := p;
3    repeat
4            for j := 1 to n − 1 do                                        /* get locks */
5                if b.lock[j] = 0 then b.lock[j] := p fi od
6            locked := 1;
7            for j := 1 to n − 1 do                                  /* have all locks? */
8                if b.lock[j] ≠ p then locked := 0 fi od;
9    until b.turn ≠ p or locked = 1 or b.winner = 1;
10   if b.turn ≠ p or b.winner = 1 then
11           for j := 1 to n − 1 do                            /* lost, release locks */
12               if b.lock[j] = p then b.lock[j] := 0 fi od
13           return(0) fi;
14   b.winner := 1; return(1).                                               /* wins */
end_function

function reset (b:bit);                                                  /* access bit b */
1    b.winner := 0; b.turn := 0;                                       /* release locks */
2    for j := 1 to n − 1 do
3            if b.lock[j] = p then b.lock[j] := 0 fi od.
end_function
```

In the test-and-set operation, a process, say p, initially checks whether $b.turn \neq 0$, and if so returns 0. Otherwise, p takes priority by setting $b.turn$ to p, and attempts to obtain all the $n-1$ locks by setting them to p. This prevents other processes that also saw $b.turn = 0$ and set $b.turn$ to their ids from entering. That is, if p obtains all the locks before the other processes set $b.turn$, they will not be able to get any of the locks since the values of the locks are not 0. Otherwise, if p sees $b.turn \neq p$ or $b.winner = 1$, it will release the locks it holds, allowing some other process to proceed, and will return 0. In the reset operation, p sets $b.turn$ to 0, so the other processes can proceed, and releases all the locks it currently holds. In [28], it is shown that even in the absence of faults, any implementation of a test-and-set bit for n processes from atomic read/write registers must use at least n such registers.

4.4 Additional results

The following additional results are presented in [28].

Election. It is known that there is no 1-resilient election algorithm using atomic registers [18, 31]. It is shown that:

> *There is an almost-wait-free symmetric election algorithm using* $\lceil \log n \rceil +$ 2 *atomic registers.*

The known space lower bound for election in the absence of faults is $\lceil \log n \rceil + 1$ atomic registers [24].

Perfect Renaming. A *perfect* renaming algorithm allows n processes with initially distinct names from a large name space to acquire distinct new names from the set $\{1, ...n\}$. A *one-shot* renaming algorithm allows each process to acquire a distinct new name just once. A *long-lived* renaming algorithm allows processes to repeatedly acquire distinct names and release them. It is shown that:

> (1) *There is a partially-wait-free symmetric one-shot perfect renaming algorithm using* (a) $n - 1$ *almost-wait-free election objects, or* (b) $O(n \log n)$ *registers.* (2) *There is a partially-wait-free symmetric long-lived perfect renaming algorithm using either* $n - 1$ *almost-wait-free test-and-set bits.*

It is known that in asynchronous systems where processes communicate by atomic registers there is no 1-resilient perfect renaming algorithm [18, 31].

Fetch-and-add, swap, stack. A *fetch-and-add* object supports an operation which takes as arguments a shared register r, and a value *val*. The value of r is incremented by *val*, and the old value of r is returned. A *swap* object supports an operation which takes as arguments a shared registers and a local register and atomically exchange their values. A shared stack is a linearizable object that supports push and pop operations, by several processes, with the usual stack semantics. It is shown that:

> *There are partially-wait-free implementations of a fetch-and-add object, a swap object, and a stack object using atomic registers.*

The result complements the results that in asynchronous systems where processes communicate using registers there are no 2-resilient implementations of fetch-and-add, swap, and stack objects [8].

Consensus and Set-consensus. The *k-set consensus* problem is to find a solution for n processes, where each process starts with an input value from some domain, and must choose some participating process' input as its output. All n processes together may choose no more than k distinct output values. The 1-set consensus problem, is the familiar consensus problem. It is shown that:

> (1) *For $n \geq 3$ and $1 \leq k \leq n-2$, there is no weakly-k-resilient k-set-consensus algorithm using either atomic registers or sending and receiving messages. In particular, for $n \geq 3$, there is no weakly-1-resilient consensus algorithm using either atomic registers or messages.* (2) *For $n \geq 3$ and $1 \leq k \leq n-2$, there is no weakly-k-resilient k-set-consensus algorithm using almost-wait-free test-and-set bits and atomic registers.*

These results strengthen the know results that, in asynchronous systems where processes communicate either by atomic registers or by sending and receiving messages, there is no 1-resilient consensus algorithm [7, 15], and there is no k-resilient k-set-consensus algorithm [3, 11, 22].

5 Related Work

All the ideas and results presented in this survey are from [27, 28, 29]. Mutual exclusion locks were first introduced by Edsger W. Dijkstra in [5]. Since than, numerous implementations of locks have been proposed [20, 25]. The fair synchronization algorithm, presented in Section 3.3, uses some ideas from the mutual exclusion algorithm presented in [26].

Algorithms for several concurrent data structures based on locking have been proposed since at least the 1970's [2]. Speculative lock elision [21], is a hardware technique which allows multiple processes to concurrently execute critical sections protected by the same lock; when misspeculation, due to data conflicts, is detected rollback is used for recovery, and the execution fall back to acquiring the lock and executing non-speculatively.

The benefits of avoiding locking has already been considered in [6]. There are many implementations of data structures which avoid locking [12, 19, 25]. Several progress conditions have been proposed for data structures which avoid locking. The most extensively studied conditions, in order of decreasing strength, are wait-freedom [8], non-blocking [13], and obstruction-freedom [9]. Progress conditions, called k-waiting, for $k \geq 0$, which capture the "amount of waiting" of processes in asynchronous concurrent algorithm, are introduced in [30].

Extensions of the notion of fault tolerance, which are different from those considered in Section 4, were proposed in[4], where a precise way is presented to characterize adversaries by introducing the notion of disagreement power: the biggest integer k for which the adversary can prevent processes from agreeing on k values when using registers only; and it is shown how to compute the disagreement power of an adversary.

Linearizability is defined in [13]. A tutorial on memory consistency models can be found in [1]. Transactional memory is a methodology which has gained

momentum in recent years as a simple way for writing concurrent programs [10, 12, 23]. It has implementations that use locks and others that avoid locking, but in both cases the complexity is hidden from the programmer.

6 Discussion

None of the known synchronization techniques is optimal in all cases. Despite the known weaknesses of locking and the many attempts to replace it, locking still predominates. There might still be hope for a “silver bullet", but until then, it would be constructive to also consider integration of different techniques in order to gain the benefit of their combined strengths. Such integration may involve using a *mixture* of objects which avoid locking together with lock-based objects; and, as suggested in Section 2, *fusing* lockless objects and locks together in order to create new interesting types of shared objects.

In Section 3, we have proposed to enforce fairness as a wrapper around a concurrent data structure, and studied the consequences. We have formalized the fair synchronization problem, presented a solution, and then showed that existing concurrent data structures and mutual exclusion algorithms can be encapsulated into a fair synchronization construct to yield algorithms that are inherently fair. Since many processes may enter their fair sections simultaneously, it is expected that using fair synchronization algorithms will not degrade the performance of concurrent applications as much as locks. However, as in the case of using locks, slow or stopped processes may prevent other processes from ever accessing their fair sections.

Finally, in Section 4, we have refined the traditional notion of *t-resiliency* by defining the finer grained notion of (t, f)-*resiliency*. Rather surprisingly, while some problems, that have no solutions which can tolerate even a single fault, do have solutions which satisfy almost-wait-freedom, other problems do not even have weakly-1-resilient solutions.

References

[1] S. V. Adve and K. Gharachorloo. Shared memory consistency models: A tutorial. *IEEE Computer*, 29(12):66–76, 1996.

[2] R. Bayer and M. Schkolnick. Concurrency of operations on B-trees. *Acta Informatica*, 9:1–21, 1977.

[3] E. Borowsky and E. Gafni. Generalizecl FLP impossibility result for *t*-resilient asynchronous computations. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 91–100, 1993.

[4] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and A. Tielmanns. The disagreement power of an adversary. In *Proc. 28th ACM Symp. on Principles of Distributed Computing*, pages 288–289, 2009.

[5] E. W. Dijkstra. Solution of a problem in concurrent programming control. *Communications of the ACM*, 8(9):569, 1965.

[6] W. B. Easton. Process synchronization without long-term interlock. In *Proc. of the 3rd ACM symp. on Operating systems principles*, pages 95–100, 1971.

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

[8] M. P. Herlihy. Wait-free synchronization. *ACM Trans. on Programming Languages and Systems*, 13(1):124–149, January 1991.

[9] M. P. Herlihy, V. Luchangco, and M. Moir. Obstruction-free synchronization: Double-ended queues as an example. In *Proc. of the 23rd International Conference on Distributed Computing Systems*, page 522, 2003.

[10] M. P. Herlihy and J.E.B. Moss. Transactional memory: architectural support for lock-free data structures. In *Proc. of the 20th annual international symposium on Computer architecture*, pages 289–300, 1993.

[11] M. P. Herlihy and N. Shavit. The topological structure of asynchronous computability. *Journal of the ACM*, 46(6):858–923, July 1999.

[12] M. Herlihy and N. Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufmann Publishers, 2008. 508 pages.

[13] M. P. Herlihy and J. M. Wing. Linearizability: a correctness condition for concurrent objects. *toplas*, 12(3):463–492, 1990.

[14] L. Lamport. A fast mutual exclusion algorithm. *ACM Trans. on Computer Systems*, 5(1):1–11, 1987.

[15] M.C. Loui and H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, 4:163–183, 1987.

[16] M. Moir and J. H. Anderson. Wait-free algorithms for fast, long-lived renaming. *Science of Computer Programming*, 25(1):1–39, October 1995.

[17] M. Merritt and G. Taubenfeld. Computing with infinitely many processes. *Information and Computation* 233 (2013) 12–31. (Also in: LNCS 1914 Springer Verlag 2000, 164–178, DISC 2000.)

[18] S. Moran and Y. Wolfstahl. Extended impossibility results for asynchronous complete networks. *Information Processing Letters*, 26(3):145–151, 1987.

[19] M. Raynal. *Concurrent Programming: Algorithms, Principles, and Foundations*. Springer. ISBN 978-3-642-32027-9, 515 pages, 2013.

[20] M. Raynal. *Algorithms for mutual exclusion*. The MIT Press, 1986. Translation of: Algorithmique du parallélisme, 1984.

[21] R. Rajwar and J. R. Goodman, Speculative Lock Elision: Enabling Highly Concurrent Multithreaded Execution. In *Proc. 34th Inter. Symp. on Microarchitecture*, pp. 294–305, 2001.

[22] M. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM Journal on Computing*, 29, 2000.

[23] N. Shavit and D. Touitou. Software transactional memory. In *Proc. 14th ACM Symp. on Principles of Distributed Computing*, pages 204–213, 1995.

[24] E. Styer and G. L. Peterson. Tight bounds for shared memory symmetric mutual exclusion problems. In *Proc. 8th ACM Symp. on Principles of Distributed Computing*, pages 177–191, August 1989.

[25] G. Taubenfeld. *Synchronization Algorithms and Concurrent Programming*. Pearson / Prentice-Hall. ISBN 0-131-97259-6, 423 pages, 2006.

[26] G. Taubenfeld. The black-white bakery algorithm. In *18th international symposium on distributed computing*, October 2004. *LNCS 3274* Springer Verlag 2004, 56–70.

[27] G. Taubenfeld. Contention-sensitive data structures and algorithms. In *23rd international symposium on distributed computing*, September 2009. *LNCS 5805* Springer Verlag 2009, 157–171.

[28] G. Taubenfeld. A closer look at fault tolerance. In *Proc. 31st ACM Symp. on Principles of Distributed Computing*, pages 261–270, 2012.

[29] G. Taubenfeld. Fair synchronization. In *27th international symposium on distributed computing*, October 2013. *LNCS 8205* Springer Verlag 2013, 179–193.

[30] G. Taubenfeld. Waiting without locking. Unpublished manuscript, 2014.

[31] G. Taubenfeld and S. Moran. Possibility and impossibility results in a shared memory environment. *Acta Informatica*, 33(1):1–20, 1996.

Local Coordination and Symmetry Breaking

Jukka Suomela
Helsinki Institute for Information Technology HIIT,
Department of Information and Computer Science,
Aalto University, Finland · jukka.suomela@aalto.fi

Abstract

This article gives a short survey of recent *lower bounds* for *distributed graph algorithms*. There are many classical graph problems (e.g., maximal matching) that can be solved in $O(\Delta + \log^* n)$ or $O(\Delta)$ communication rounds, where n is the number of nodes and Δ is the maximum degree of the graph. In these algorithms, the key bottleneck seems to be a form of *local coordination*, which gives rise to the linear-in-Δ term in the running time. Previously it has not been known if this linear dependence is necessary, but now we can prove that there are graph problems that can be solved in time $O(\Delta)$ independently of n, and cannot be solved in time $o(\Delta)$ independently of n. We will give an informal overview of the techniques that can be used to prove such lower bounds, and we will also propose a roadmap for future research, with the aim of resolving some of the major open questions of the field.

1 Introduction

The research area of distributed computing studies computation in large computer networks. The fundamental research question is *what can be computed efficiently in a distributed system*. In distributed systems, communication is several orders of magnitude slower than computation: while a modern computer can perform arithmetic operations in a matter of *nanoseconds*, it can easily take dozens of *milliseconds* to exchange messages between two computers over the public Internet. To understand which computational tasks can be solved efficiently in a computer network, it is necessary to understand what can be solved with *very few communication steps*.

1.1 Model of Computing

Perhaps the most successful theoretical model for studying such questions is the LOCAL model [25,32]: We have an unknown graph G that represents a computer network. Every node of G is a computer and every edge of G is a communication link. Initially, each computer is only aware of its immediate surroundings. Computation proceeds in synchronous rounds; in each round, all nodes exchange messages with their neighbours. Eventually, all nodes have to stop and announce their local outputs—that is, their own part of the solution. For example, if we are studying graph colouring, each node has to output its own colour.

The *distributed time complexity* of a graph problem is the smallest t such that the problem can be solved with a distributed algorithm in t communication rounds. In the LOCAL model, parameter t plays a dual role—it represents both *time* and *distance*: in t rounds, all nodes can learn everything about graph G in their radius-t neighbourhood, and nothing else. In essence, a distributed algorithm with a running time of t is a mapping from radius-t neighbourhoods to local outputs—see Figure 1 for an illustration.

Therefore the defining property of fast distributed algorithms is *locality*: in a fast distributed algorithm each node only needs information from its local neighbourhood. In many cases, if we are given a graph problem, it is fairly easy to qualitatively classify it as a "local" or "global" problem; the key challenge is to understand precisely *how* local a given problem is.

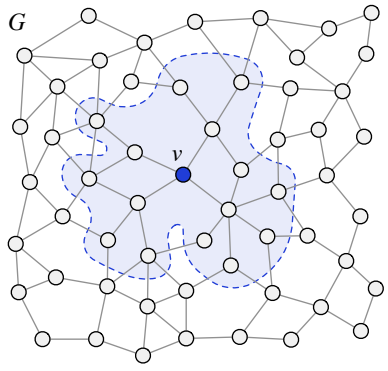


Figure 1: If we have a distributed algorithm with a running time of $t = 2$ rounds, then whatever node v outputs is a function of the information available in its radius-t neighbourhood (highlighted).

1.2 Symmetry Breaking vs. Local Coordination

Distributed time complexity and locality are commonly studied as a function of two parameters:

- n, the number of nodes in the network,
- Δ, the maximum degree of a node (the maximum number of neighbours).

As we will see, these parameters are often related to two different challenges in the area of distributed algorithms:

1. complexity as a function of n is related to *symmetry breaking*,
2. complexity as a function of Δ is related to *local coordination*.

The first aspect has been relatively well understood since Linial's seminal work in the 1990s [25]. However, our understanding of the second aspect has been poor—until very recently.

In this article, we will study the challenge of local coordination in more depth. We will survey recent work that has enabled us to better understand the distributed time complexity of certain graph problems not only as a function of n, but also as a function of Δ. Hopefully these preliminary results will provide a starting point for a research program that aims at resolving the long-standing open questions related to the distributed time complexity of classical graph problems such as maximal matchings, maximal independent sets, and graph colouring.

2 The LOCAL Model of Distributed Computing

To get started, let us first define the LOCAL model of distributed computing a bit more carefully. A reader familiar with the model can safely skip this section; further information can be found in many textbooks [27, 32, 34].

We will study distributed algorithms in the context of graph problems. We are given an unknown graph G, and the algorithm has to produce a feasible solution of the graph problem. Each node of graph G is a computational entity, and all nodes run the same distributed algorithm A. Eventually, each node v has to stop and produce its own part of the solution—the *local output* $A(G, v)$.

For example, if our goal is to find a proper vertex colouring of G, then the local output $A(G, v)$ will be the colour of node v in the solution. If our goal is to find a maximal matching, then the local output $A(G, v)$ will indicate whether v is matched and with which neighbour.

2.1 Synchronous Message Passing

In the LOCAL model, each node has a unique identifier, and initially each node knows only its own unique identifier and its degree in graph G. Computation proceeds in *synchronous communication rounds*. In every round, each node v that is still running performs the following steps, synchronously with all other nodes:

1. v sends a message to each of its neighbours,
2. v receives a message from each of its neighbours,
3. v updates its local state, and
4. possibly v stops and announces its local output $A(G, v)$.

The outgoing messages are a function of the old local state, and the new local state is a function of the old state and the messages that the node received.

The *running time* of an algorithm is defined to be the number of communication rounds until all nodes have stopped and announced their local outputs. This is the key difference between centralised and distributed computing: in the context of centralised algorithms we are interested in the number of *elementary computational steps*, while in the context of distributed algorithms the key resource is the number of *communication steps*. For our purposes, the cost of local computation is negligible.

2.2 Time and Distances Are Equivalent

In the LOCAL model, information can propagate at a maximum speed of 1 edge per round. If a node v stops after t rounds, then whatever v outputs can only depend on the information that was available within distance t from v in the input graph. In essence, a time-t algorithm in the LOCAL model is just a *mapping from radius-t neighbourhoods to local outputs*; recall Figure 1.

A bit more formally, let us write $x_t(v)$ for the local state of a node v after round t. Initially, $x_0(v)$ consists of just the unique identifier of node v and its degree. The key observation is that $x_{t+1}(v)$ is a function of $x_t(v)$ and the messages that v received from its neighbours during round $t + 1$. For each neighbour u of v, the message sent by u to v during round $t + 1$ is a function of $x_t(u)$. Hence the new state $x_{t+1}(v)$ is simply a function of the old states $x_t(v)$ and $x_t(u)$, where u ranges over the neighbours of v.

In essence, in each communication round, each node just updates its local state based on the local states in its radius-1 neighbourhood. By a simple induction, the local state $x_t(v)$ is a function of the initial states $x_0(u)$, where u ranges over all nodes that are within distance t from v. Hence we can see that if a node stops after round t, its local output can only depend on the information that was available within distance t from v.

Conversely, it is easy to design an algorithm in which all nodes can gather their radius-t neighbourhoods in t communication rounds. As a corollary, if graph G is connected and has a diameter of D, then in time $t = D + O(1)$ all nodes can learn everything about the structure of the input graph G. Therefore in time $t = D + O(1)$ we can also solve any computable graph problem: each node can simply gather full information on the input, and then locally solve the problem and output the relevant part of the solution. The diameter is at most $O(n)$, and therefore linear-time algorithms are entirely trivial in the LOCAL model; the research on the LOCAL model focuses on algorithms that run in sublinear time.

3 Example: Maximal Matchings

As a concrete example, let us consider the problem of finding a *maximal matching* with a distributed algorithm in the LOCAL model. Recall that a *matching* of a simple undirected graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that each node is incident to at most one edge of M, and a matching is *maximal* if it is not a proper subset of another matching.

3.1 Simple Distributed Algorithms

It is easy to come up with a distributed algorithm that finds a maximal matching. Here is a simple example:

- Initialise $M \leftarrow \emptyset$.
- Use the unique node identifiers to assign a unique label for each edge.
- Repeat until G is empty:
  - Find the set $X \subseteq E$ that consists of the edges that are *local minima* with respect to the edge labels, i.e., their labels are smaller than the labels of any of their neighbours.
  - Add X to M, and remove all edges adjacent to X from G.

Unfortunately, the running time of such an algorithm can be linear in n, which makes it uninteresting from the perspective of the LOCAL model.

In the above algorithm, a key drawback is that set X can be very small in the worst case. We can do better with a simple *randomised* distributed algorithm [1, 20, 26]. Instead of picking local minima, we can pick a random matching X. More precisely, each edge e attempts to join X with a certain (carefully chosen) probability, and if none of its neighbours attempts to join simultaneously, we will

have $e \in X$. The basic idea is that we can eliminate a constant fraction of the edges in each step, and it can be shown that the running time will be $O(\log n)$ with high probability.

3.2 State of the Art

The fastest distributed algorithms can find maximal matchings much faster than in logarithmic time—at least in sparse graphs. To better characterise the running time, we will use two parameters: n, the number of nodes, and Δ, the maximum degree of the graph. As a function of n and Δ, currently the fastest algorithms are these:

- Barenboim et al. [7]: a randomised algorithm, time $O(\log \Delta + \log^4 \log n)$.
- Hańćkowiak et al. [18]: a deterministic algorithm, time $O(\log^4 n)$.
- Panconesi and Rizzi [31]: a deterministic algorithm, time $O(\Delta + \log^* n)$.

Here $\log^* n$ is the iterated logarithm of n, a very slowly growing function.

3.3 Focus on Low-Degree Graphs

In general, charting the landscape of distributed time complexity throughout the (n, Δ)-plane is an arduous task. In this article, we will zoom into one corner of the plane: we will focus on the case of $n \gg \Delta$.

In this region, the Panconesi–Rizzi algorithm, which runs in time $O(\Delta + \log^* n)$, is the fastest known algorithm for finding maximal matchings. While the expression of the running time may not look particularly natural, it turns out there are many other problems for which the fastest algorithms in sparse graphs have a running time of precisely $O(\Delta + \log^* n)$—the key examples are maximal independent sets, vertex colouring, and edge colouring [5, 6, 21, 31].

While many different algorithms with precisely the same running time exist, we do not know if any of these are optimal. In particular, it is not known if any of these problems could be solved in time $o(\Delta) + O(\log^* n)$.

In what follows, we will dissect the running time of the Panconesi–Rizzi algorithm in two terms, $O(\log^* n)$ and $O(\Delta)$, and give an intuitive explanation for each of them. To do this, we will consider two corner cases: one in which we can find maximal matchings in time $O(\log^* n)$, and one in which the problem can be solved in time $O(\Delta)$. As we will see, there is a very good justification for the term $O(\log^* n)$, but there is no lower bound that would justify the existence of the term $O(\Delta)$.

3.4 Maximal Matchings as a Symmetry-Breaking Problem

First, let us have a look at the term $O(\log^* n)$. Maximal matchings are fundamentally a *symmetry-breaking problem*. To see this more clearly, we will define a restricted version in which we are left with a *pure* symmetry-breaking problem; in essence, we will eliminate the term Δ from the running time.

This turns out to be easy: we can simply focus on cycle graphs, in which case we have a constant maximum degree of $\Delta = 2$. We are now left with the seemingly trivial problem of finding a maximal matching in a cycle.

This special case highlights the need for symmetry breaking. While the topology of the input graph is symmetric, the output cannot be symmetric: any maximal matching has to contain some of the edges, but not all of them. We will have to resort to some means of symmetry breaking.

Problems of this type can be solved efficiently with the help of the Cole–Vishkin algorithm [8]. This is a deterministic distributed algorithm that can be used to e.g. colour a cycle with $O(1)$ colours in time $O(\log^* n)$. The algorithm relies heavily on the existence of unique node identifiers; it extracts symmetry-breaking information from the unique identifiers.

In essence, the Cole–Vishkin algorithm is a *colour reduction algorithm*. The unique identifiers provide a colouring with a large number of colours. Typically it is assumed that the range of unique identifiers is polynomial in n, and hence our starting point is a poly(n)-colouring of the cycle. The Cole–Vishkin algorithm then reduces the number of colours from any number ℓ to $O(\log \ell)$ in one step; it compares the *binary representations* of the old colours of adjacent nodes, and uses the index of the bit that differs, together with its value, to construct a new colour. Overall, we need only $O(\log^* n)$ iterations to reduce the number of colours to $O(1)$.

Once we have a colouring of the vertices with $O(1)$ colours, it is easy to find a maximal matching. For example, we can use the vertex colouring to derive an edge colouring. Then each colour class is a matching, and we can construct a maximal matching by considering the colour classes one by one. Hence the pure symmetry-breaking version of maximal matchings can be solved in time $O(\log^* n)$.

Remarkably, this is also known to be optimal. Linial's seminal lower bound [25] shows that symmetry breaking in a cycle requires $\Omega(\log^* n)$ rounds with deterministic algorithms, and Naor [28] shows that this holds even if we consider randomised algorithms.

In summary, pure symmetry breaking problems are very well understood. As a simple corollary, it is not possible to find a maximal matching in time $O(\Delta) + o(\log^* n)$, or in time $f(\Delta) + o(\log^* n)$ for any function f.

3.5 Maximal Matchings as a Coordination Problem

Let us now turn our attention to the term $O(\Delta)$. We will argue that maximal matchings are also a *coordination problem*. To see this more clearly, we will again define a restricted version in which we are left with a *pure* coordination problem, which can be solved in time $O(\Delta)$ independently of n.

Of course we cannot simply set $n = O(1)$ to get rid of the term $O(\log^* n)$. However, we can consider a situation in which we have already solved the symmetry-breaking problem. We will *assume* that we are given a *bipartite graph*, in which one part is coloured black, another part is white. Our goal is to find a maximal matching in such a graph.

In essence, each black node should try to pick one of its white neighbours as its partner, and conversely each white node should try to pick one of its black neighbours as its partner. Here we have the coordination challenge: without any coordination, several black nodes may try to pick the same white node simultaneously.

In bipartite graphs, this coordination problem can be solved in time $O(\Delta)$ with a simple proposal algorithm [17]. The algorithm repeatedly performs the following steps:

1. Black nodes send "proposals" to their white neighbours, one by one (e.g. in the order of their unique identifiers).

2. White nodes "accept" the first proposal that they get (breaking ties with e.g. the unique identifiers of the senders), and "reject" all other proposals.

Each proposal–acceptance pair forms an edge in the matching. A black node will stop as soon as it becomes matched, or it runs out of white neighbours to whom to send proposals (in which case all of the neighbours are already matched). A white node will stop as soon as it becomes matched, or all of its black neighbours have stopped (in which case all of the neighbours are already matched). Clearly the output is a maximal matching.

While the simple proposal algorithm runs in time $O(\Delta)$ independently of n, it is not known if it is optimal. More specifically, it is not known if we can find a maximal matching in bipartite 2-coloured graphs in time $o(\Delta)$ independently of n. However, this is the best algorithm that we currently have for this problem; we have not been able to break the linear-in-Δ boundary (without introducing some dependence on n in the running time). Many other distributed algorithms have a similar issue at their core: in one way or another, there is a need for local coordination, and this is resolved in a fairly naive manner by considering neighbours one by one.

While symmetry breaking is well understood, local coordination is poorly understood. There are hardly any lower bounds. For bipartite maximal matchings

we can apply the lower bound by Kuhn et al. [22–24], which shows that bipartite maximal matching requires $\Omega(\log \Delta)$ rounds in the worst case. However, this still leaves an *exponential* gap between the upper bound and the lower bound.

A particularly intriguing special case is that of *regular graphs*, i.e., the case that all nodes have the same degree of Δ. In this case, the simple proposal algorithm is still the fastest algorithm that we have, and hence the best known upper bound is still $O(\Delta)$. Somewhat surprisingly, there are no lower bounds at all for this case; the above-mentioned lower bound by Kuhn et al. cannot be applied here any more. Hence we have very simple coordination problems whose distributed time complexity is not understood at all.

4 Towards Coordination Lower Bounds

We have seen that in low-degree graphs, the fastest algorithm for maximal matchings has a running time of $O(\Delta + \log^* n)$. Here the term $O(\log^* n)$ is related to the well-known challenge of symmetry breaking, while the term $O(\Delta)$ appears to be related to a poorly-understood challenge of local coordination. The problem cannot be solved in time $O(\Delta) + o(\log^* n)$, but it is a major open question if it can be solved in time $o(\Delta) + O(\log^* n)$. Let us now have a look at possible ways towards answering this question.

4.1 Completable but Tight Problems

As we have discussed, maximal matchings are not an isolated example. There are numerous other graph problems for which the time complexity as a function of n is well-understood (at least for a small Δ), but the time complexity as a function of Δ is not really understood at all.

Perhaps the most prominent example is proper vertex colouring with $\Delta + 1$ colours [5, 6, 21]. This is yet another problem that can be solved in $O(\Delta + \log^* n)$ time, and it is known to require $\Omega(\log^* n)$ time, but it is not known if the problem can be solved in $o(\Delta) + O(\log^* n)$ time.

Other examples of such problems include edge colouring with $2\Delta - 1$ colours, and the problem of finding a maximal independent set. Incidentally, all of these problems can be characterised informally as follows:

1. Any partial solution can be *completed*. In particular, a greedy algorithm that considers nodes or edges in an arbitrary order can solve these problems.

2. However, there are situations in which a partial solution is *tight* in the sense that there is only one way to complete it. For example, if we have already

coloured all Δ neighbours of a node, and we have a colour palette of size $\Delta + 1$, we may have only 1 possible colour left.

While some counterexamples exist, it seems that many problems of this form have distributed algorithms with a running time of, e.g., $O(\Delta + \log^* n)$ or simply $O(\Delta)$, and we do not know if these algorithms are optimal. However, as soon as we step outside the realm of "completable but tight" problems, we see plenty of examples of running times that are either sublinear or superlinear in Δ:

1. If we drop the first restriction, running times typically increase to *superlinear* in Δ. Examples of such problems include matchings without short augmenting paths [2], which can be solved in time $\Delta^{O(1)}$, and locally optimal semi-matchings [9], which can be solved in time $O(\Delta^5)$.

2. If we drop the second restriction, there are many problems that can be solved in time *sublinear* in Δ. A good example is vertex colouring with $O(\Delta^2)$ colours. With such a large colour palette, local coordination becomes much easier: Linial's algorithm [25] solves this problem in time $O(\log^* n)$, independently of Δ.

Hence to resolve the long-standing open questions related to the distributed time complexity, it seems that we need to better understand the issue of local coordination in the context of "completable but tight" problems.

4.2 Major Hurdles and Recent Advances

Looking back at the previous attempts to prove e.g. tight lower bounds for the distributed time complexity of maximal matchings, it now seems that one of the major hurdles can be summarised as follows:

1. We do not understand the issue of local coordination even in isolation.

2. To prove tight lower bounds, we would need to understand not just local coordination in isolation, but also the complicated interplay of local coordination and symmetry breaking.

While the second issue seems to be still far beyond the reach of current research, we are now finally making progress with the first issue. The key idea is to identify *pure coordination problems*. These are "completable but tight" problems that can be solved in time $O(\Delta)$ independently of n, but cannot be solved in time $o(\Delta)$ independently of n. Such problems do not need any symmetry breaking, and hence we can focus solely on local coordination.

Now we finally have the first natural example of a pure coordination problem for which we can prove tight lower bounds: *maximal fractional matching*. This

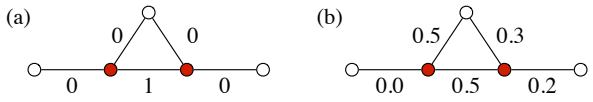


Figure 2: (a) A maximal matching. (b) A maximal fractional matching. The saturated nodes are highlighted.

was known to be solvable in time $O(\Delta)$ independently of n [3], and there is now a lower bound that shows that the problem cannot be solved in time $o(\Delta)$ independently of n [14]. In the next section, we will discuss this problem in more depth.

5 Recent Progress: Maximal Fractional Matchings

A *fractional matching* is a linear programming relaxation of a matching. While maximal matchings are a symmetry-breaking problem, it turns out that the problem of finding a *maximal fractional matching* is a pure coordination problem—it does not require any symmetry breaking.

5.1 Problem Formulation

If a matching can be interpreted as an integer-valued function $y\colon E \to \{0,1\}$ that indicates which edges are in the matching, a fractional matching is a real-valued function $y\colon E \to [0,1]$.

Formally, let $G = (V,E)$ be a simple undirected graph and let $y\colon E \to [0,1]$ associate weights to the edges of G. Define for each $v \in V$ the sum of incident edges

$$y[v] := \sum_{e \in E : v \in e} y(e).$$

The function y is called a *fractional matching* if $y[v] \leq 1$ for each node v. A node v is *saturated* if $y[v] = 1$. A fractional matching is *maximal* if each edge has at least one saturated endpoint.

Informally, in a maximal fractional matching we cannot increase the weight of any edge without violating a constraint. See Figure 2 for examples.

In combinatorial optimisation, maximal fractional matchings and maximal matchings have similar applications. It is well known that we can use a maximal matching to construct a 2-approximation of a minimum vertex cover. It turns out that we can use maximal fractional matchings equally well: in any maximal fractional matching, the set of saturated nodes forms a 2-approximation of a minimum vertex cover [4].

5.2 No Need for Symmetry Breaking

From the perspective of centralised algorithms, the distinction between maximal matchings and maximal fractional matchings is not particularly interesting; either of the problems can be solved easily in linear time. However, the two problems are very different from the perspective of efficient distributed or parallel algorithms.

While the problem of finding a maximal matching requires symmetry breaking (recall Section 3.4), this is not the case with maximal fractional matchings. Consider, for example, a k-regular graph. In any such graph, there is a trivial maximal fractional matching that sets $y(e) = 1/k$ for all $e \in E$. In essence, highly symmetric graphs are trivial from the perspective of maximal fractional matchings; only non-symmetric inputs require some effort.

The general case is not that easy to solve efficiently, but it turns out that there is an algorithm [3] that finds a maximal fractional matching in $O(\Delta)$ time in the LOCAL model. The running time does not have any dependence on n, but it is still linear in Δ, as the algorithm resorts to a fairly naive one-by-one approach to overcome challenges related to local coordination.

In summary, we can characterise the state-of-the-art algorithms for these two problems as follows:

1. Maximal matchings:
   - symmetry breaking necessary
   - algorithms resort to local coordination
   - time $O(\Delta + \log^* n)$.
2. Maximal fractional matchings:
   - symmetry breaking not needed
   - algorithms resort to local coordination
   - time $O(\Delta)$.

5.3 A New Lower Bound

Maximal fractional matchings are a genuine example of a pure coordination problem. They are a "completable but tight" problem, in the sense discussed in Section 4.1. They do not need any symmetry breaking. Most importantly, now we can show that the $O(\Delta)$-time algorithm is optimal (at least for sparse graphs): the problem cannot be solved in time $o(\Delta)$, independently of n, with any distributed algorithm (deterministic or randomised).

Before we continue, it is important to emphasise that the result does not yet tell anything more than what is stated above. In particular, it has not yet been

ruled out that there could exist a sublinear-in-Δ algorithm whose running time depends moderately on n. For example, algorithms of a running time $o(\Delta) + O(\log^* n)$ cannot be yet excluded. This is also the reason why this result does not tell anything interesting about maximal matchings: a simple corollary would be that maximal matchings cannot be found in time $o(\Delta)$ independently of n, but this we already know, as any algorithm for maximal matchings has a running time $\Omega(\log^* n)$ that certainly depends on n. Nevertheless, this result demonstrates that there are techniques with which we can prove tight lower bounds for pure coordination problems, and this hopefully paves the road for future research in which we can tackle also algorithms with running times that have a moderate dependence on n.

A key insight in the proof is that we will not try to prove the result directly for the LOCAL model, but we will first consider *weaker* models of distributed computing (Sections 5.4 and 5.7). In weaker models, lower bounds are of course easier to prove but less interesting. Only then we will amplify the result so that we have similar lower bounds also for the LOCAL model.

On a high level, the proof builds on the following techniques:

1. The *unfold-and-mix* technique for proving lower bounds in very weak models of distributed computing (Section 5.6).

2. A general technique for amplifying lower bounds from weak models to the usual LOCAL model (Section 5.8). The main ingredient here is the construction of so-called *homogeneous graphs* (Section 5.9).

Both of these techniques were originally presented in PODC 2012 [13, 19], but it was not until PODC 2014 [14] that we managed to put these techniques together in order to prove lower bounds for the maximal fractional matching problem in the usual LOCAL model.

5.4 Port-Numbering Model and Edge Colouring Model

We start by introducing two models of distributed computing: the *port-numbering model*, in short PN, and the *edge-colouring model*, in short EC. While at least the PN model is also interesting in its own right, for our purposes these models serve as a stepping stone towards more interesting results—lower bounds in the LOCAL model.

In many ways, the PN and EC models are similar to the usual LOCAL model: Each node is an autonomous entity that maintains its own local state. Computation proceeds in synchronous rounds. In each round, all nodes in parallel (1) send messages to their neighbours, (2) receive messages from their neighbours, (3) update their local state, and (4) possibly announce their local output and stop.

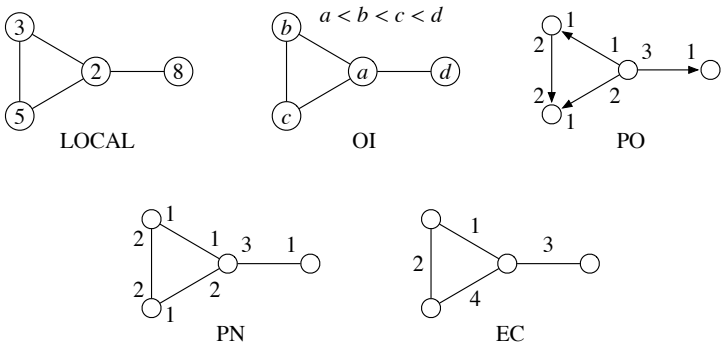
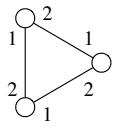


Figure 3: Models LOCAL, OI, PO, PN, and EC.

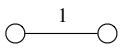
The key difference is related to the initial information that is available in the network. In PN and EC, the nodes are *anonymous*—they do not have any unique identifiers. However, each node can still distinguish between their neighbours; see Figure 3:

- In the PN model, the endpoints of the edges ("ports") are numbered so that a node of degree d is incident to endpoints with numbers $1, 2, \ldots, d$. In essence, a node can refer to its neighbours by numbers $1, 2, \ldots, d$.
- In the EC model, the edges are properly coloured with $O(\Delta)$ colours. Each node knows the colours of the incident edges, and it can use the colours to refer to its neighbours.

Note that EC is strictly stronger than PN. Given an edge colouring, it is easy to derive a port numbering. The converse is not true: for example, in the EC model it is trivial to find an edge colouring, but in the PN model it is not possible in general with any deterministic algorithm. To see this, consider a cycle with a symmetric port numbering; no PN-algorithm can break the symmetry here:



However, note that EC is also a fairly weak model. For example, no deterministic algorithm can break the symmetry in a graph with just two nodes:



In particular, it is not possible to find a proper vertex colouring with either PN or EC algorithms here. While everything was solvable in linear time in the LOCAL model, there are numerous seemingly trivial problems that cannot be solved at all in PN or EC.

5.5 Maximal Matching in the EC Model

Maximal matching cannot be solved with deterministic algorithms in the PN model. However, in the EC model, there is a very simple greedy algorithm for finding a maximal matching M in time $O(\Delta)$. We consider each colour class $1, 2, \ldots, O(\Delta)$ one by one. In step i, we find all edges of colour i that are not yet adjacent to any edge of M, and add them to M. Clearly, the end result is a maximal matching.

A key observation at this point is that maximal matchings in the EC model do not need any symmetry breaking. Symmetry between adjacent edges is already broken by the edge colouring. We only need to deal with local coordination, in order to avoid adding two adjacent edges simultaneously to M.

The greedy algorithm solves the local coordination challenge by a very naive technique: it considers colour classes one by one, which results in a linear-in-Δ running time. The key question is now if this algorithm is optimal in the EC model. Perhaps e.g. a clever divide-and-conquer approach could solve the problem in only $O(\log \Delta)$ time?

It turns out the greedy algorithm is indeed optimal. We can show that there is no algorithm that finds a maximal matching in $o(\Delta)$ time in the EC model [19]. This was the first linear-in-Δ lower bound for a natural coordination problem.

5.6 Unfold and Mix

We will now give an overview of the techniques that can be used to prove lower bounds for the maximal matching problem in the EC model. Assume that we have a deterministic distributed algorithm A that finds a maximal matching in any given graph, for any given edge colouring. With this knowledge, we can construct "fragile" instances in which algorithm A is forced to produce *perfect matchings*. Informally, perfect matchings are more tightly constrained than maximal matchings; minor changes in the input may cause major changes in the output.

To force algorithm A to produce a perfect matching, we will to study graphs with self-loops. For our purposes, a graph G with self-loops is just a compact representation of a large (possibly infinite) graph G' that does not have any loops. To construct G', we just "unfold" all loops of G; see Figure 4.

Each self-loop represents *symmetry*. If e is a self-loop in graph G, and we unfold e to construct graph G', then G' will be symmetric with respect to e. More precisely, in the EC model deterministic algorithms cannot distinguish between

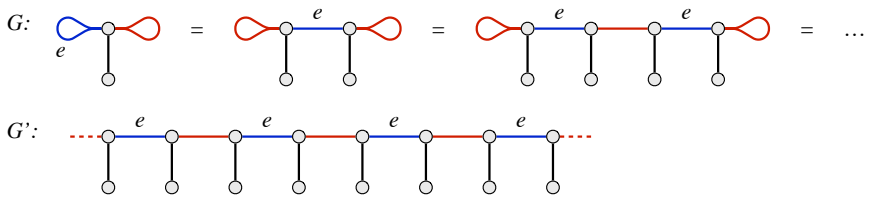


Figure 4: Graph G with self-loops is a compact representation of graph G'.

the two endpoints of e. If one of the endpoints produces the local output indicating that it is unmatched, the other endpoint will produce the same output—but this is a contradiction, as in a maximal matching we cannot have a pair of adjacent nodes such that both of them are unmatched. Therefore all nodes that have self-loops must be always matched. As long as we have at least one self-loop attached to each node, the output will be a perfect matching.

To prove the lower bound of $\Omega(\Delta)$, we will study *critical pairs*. We say that G and H form an r-critical pair of graphs if:

- G has a self-loop e and H has a self-loop f,
- loops e and f have the same colour,
- the radius-r neighbourhoods of e and f are isomorphic,
- algorithm A makes a different decision for e and f.

By a different decision we mean that in graph G algorithm A outputs a matching $A(G)$ with $e \in A(G)$, and in graph H algorithm A outputs a matching $A(H)$ with $f \notin A(H)$. Naturally, if such a pair exists, then the running time of A has to be at least r; otherwise A would not be able to distinguish between e and f.

For any algorithm A, it is fairly straightforward to find a 0-critical pair that consists of just a pair of nodes so that both of them have $\Theta(\Delta)$ self-loops. Once we have a 0-critical pair, we will apply a technique called *unfold-and-mix* repeatedly. In each iteration we will lose some self-loops but gain criticality. We can repeat the process for $\Theta(\Delta)$ times until we run out of self-loops. The end result will be a $\Theta(\Delta)$-critical pair of graphs of maximum degree Δ. The existence of such a pair is enough to demonstrate that the running time of algorithm A is $\Omega(\Delta)$.

The unfold-and-mix technique is illustrated in Figure 5. In each inductive step, our starting point is a k-critical pair of graphs, G and H, with the special loops e and f. Then we

1. "unfold" the loops e and f to obtain graphs GG and HH,
2. "mix" the graphs GG and HH together to obtain another graph GH that combines elements from G and H.

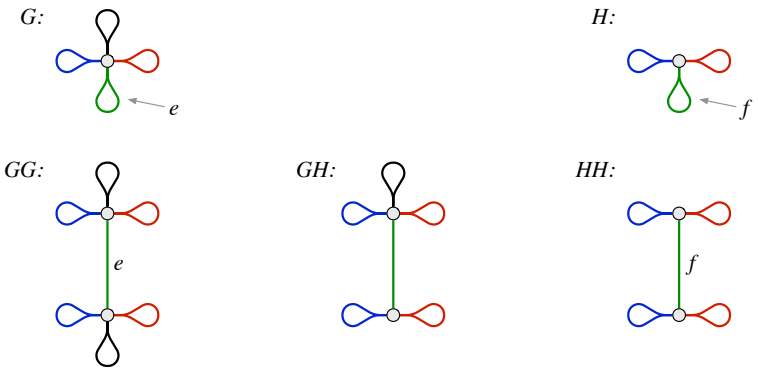


Figure 5: The unfold-and-mix technique.

Intuitively, the output of A in GG and the output of A in HH are not compatible with each other—by assumption, the outputs of e and f are different. Hence the algorithm is now in trouble:

- It has to do something different in GH in comparison with what it did in either GG or HH.
- It cannot sweep the problem under the rug easily, as the instances are "fragile": graph GH still has self-loops, and therefore the algorithm has to output a perfect matching.

With some effort, we can now show that among the three graphs that we have constructed (GG, GH, and HH), we can always find at least two that satisfy the conditions of a $(k+1)$-critical pair.

In essence, each unfold-and-mix step makes the problem instance more difficult from the perspective of the algorithm that we study: the algorithm has to look further (i.e., spend more time) in order to distinguish the two graphs that form a critical pair. This way we can show that algorithm A cannot find a maximal matching in time $o(\Delta)$. Similar ideas can be used to prove an analogous lower bound also for maximal fractional matchings.

5.7 More Models

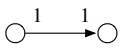
So far we have seen how to prove tight lower bounds for coordination problems in the EC model. However, we are interested in the usual LOCAL model, and the EC model and the LOCAL model are very different from each other.

We will introduce two new models that serve as intermediate steps that bridge the gap between the EC model and the LOCAL model; see Figure 3:

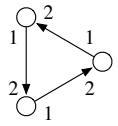
- *Port numbering and orientation* (PO): The model is a stronger version of the PN model. In addition to the port numbering, we are also given an *orientation*, i.e., for each edge one of the endpoints is labelled as the head. The orientation does not restrict how we can send information in the network; it is just additional symmetry-breaking information that the algorithm can use.
- *Order-invariant algorithms* (OI): This model is a weaker version of the LOCAL model. The nodes have unique identifiers, but algorithms can only use the relative order of the nodes and not the numerical values of the identifiers. Put otherwise, if we relabel the nodes but preserve their relative order, the output of the algorithm must not change.

For randomised algorithms these models are not that interesting, as a randomised algorithm can use random bits to e.g. generate labels that are unique with high probability (at least if we have some estimate of the size of the network). However, for deterministic algorithms the differences between the models PN, PO, OI, and LOCAL become interesting.

First, we can see that the PO model is strictly stronger than the PN model. For example, in a graph with just one edge, a PO algorithm can use the orientation to break the symmetry between the endpoints, while in PN this is not possible:



We can also see that OI is strictly stronger than PO. The OI model is clearly at least as strong as the PO model: the ordering of the nodes can be used to derive both a port numbering and an orientation. Moreover, in the OI model we can always break symmetry e.g. in a cycle (there is always a unique node that is smaller than any other node), while this is not necessarily the case in the PO model:



Finally, there are many problems that can be solved faster in the LOCAL model than in the OI model. Maximal matchings in a cycle are a good example: it takes $\Theta(n)$ time to find a maximal matching in a cycle in the OI model in the worst case, but as we have discussed in Section 3.4, this is possible in $\Theta(\log^* n)$ time in the LOCAL model. In summary, for deterministic algorithms the relative strengths of the models are PN $\subsetneq$ PO $\subsetneq$ OI $\subsetneq$ LOCAL and PN $\subsetneq$ EC.

5.8 Amplifying Lower Bounds

We have seen above that in general, the PO model can be much weaker than the LOCAL model. Indeed, there are many algorithms that exploit the properties of the LOCAL model in order to solve graph problems efficiently. For example, numerous algorithms that solve symmetry breaking in time $O(\log^* n)$ specifically rely on the unique identifiers and cannot be used in the PO model.

However, if we have a look at problems that can be solved in time $f(\Delta)$ independently of n—for example, pure coordination problems—the situation looks very different. As we can see from the survey [33], for numerous classical graph problems, the best $f(\Delta)$-time deterministic approximation algorithms in the LOCAL model do not make any use of unique identifiers. We could easily run the same algorithms in the PO model as well (and sometimes also in the PN model).

It turns out that this is not just a coincidence. We can now prove that the PO, OI, and LOCAL models are equally strong, at least in the following case [13]:

1. We use distributed algorithms with a running time of $f(\Delta)$ for some f, independently of n.

2. We are interested in so-called *simple* PO-*checkable graph optimisation problems*. This includes many classical packing and covering problems such as vertex covers, edge covers, matchings, independent sets, dominating sets, and edge dominating sets.

To prove that PO and LOCAL are equally strong for this family of problems, we proceed in two steps, using the OI model as an intermediate step:

- PO $\approx$ OI: We introduce so-called *homogeneous graphs*, in which nodes are ordered so that the ordering provides as little additional information as possible. There is only a small fraction of nodes for which OI algorithms may have an advantage over PO algorithms. See Section 5.9 for more details.

- OI $\approx$ LOCAL: We apply *Ramsey's theorem* [16] to assign unique identifiers in an unhelpful manner, so that algorithms in the LOCAL model do not have any advantage over OI algorithms.

The use of Ramsey's theorem in such a context is nowadays a standard technique [10, 29]. The key novelty is the introduction of homogeneous graphs, and in particular, a proof that shows that finite high-girth high-degree homogeneous graphs indeed exist.

5.9 Key Ingredient: Homogeneous Graphs

We introduce the following technical definition that is helpful in the context of the OI model. We say that graph G is $(1-\epsilon, r)$-homogeneous if the nodes can be ordered so that a fraction $1-\epsilon$ of all radius-r neighbourhoods are isomorphic, with respect to both topology and ordering. If we have such an ordering of the nodes, then any OI-algorithm with a running time at most r will be in trouble: almost all neighbourhoods look identical.

To show that models PO and OI are equally strong for any $f(\Delta)$-time algorithm, for a broad range of graph problems, it is desirable to have graphs with the following properties, for any g, r, k, and $\epsilon > 0$:

(1) the graph is $(1-\epsilon, r)$-homogeneous,
(2) the graph is $2k$-regular,
(3) the graph has girth at least g, and
(4) the graph is finite.

It turns out that satisfying any three out of the four properties is easy—see Figure 6 for examples:

(a) Regular high-girth graphs satisfy all properties except (1).
(b) Cycles satisfy all properties except (2).
(c) Regular grids satisfy all properties except (3).
(d) Infinite trees satisfy all properties except (4).

However, satisfying all four properties simultaneously is more challenging. The construction that we use in our recent work [15] is based on the Cayley graphs of certain groups (variants of so-called iterated wreath products) that have several desirable properties: moderate growth, relatively large girth [12], and a convenient geometric embedding.

5.10 Putting Everything Together

With the help of homogeneous graphs we have been able to show that the models PO, OI, and LOCAL are equally strong from the perspective of $f(\Delta)$-time algorithms. We also know that maximal matchings take $\Omega(\Delta)$ time in the EC model. Ideally, we would now like to put the two results together and show that maximal matchings cannot be solved in $o(\Delta) + O(\log^* n)$ time in the LOCAL model, either.

Unfortunately, we do not know how to do this yet. Even if we put aside the issue of somehow bridging the gap between the EC model and the PO model, we have a much bigger obstacle in front of us: the term $O(\log^* n)$ in the running time is enough to separate the models PO and LOCAL, and kill the argument of Section 5.8. In essence, we still cannot deal with symmetry-breaking problems.

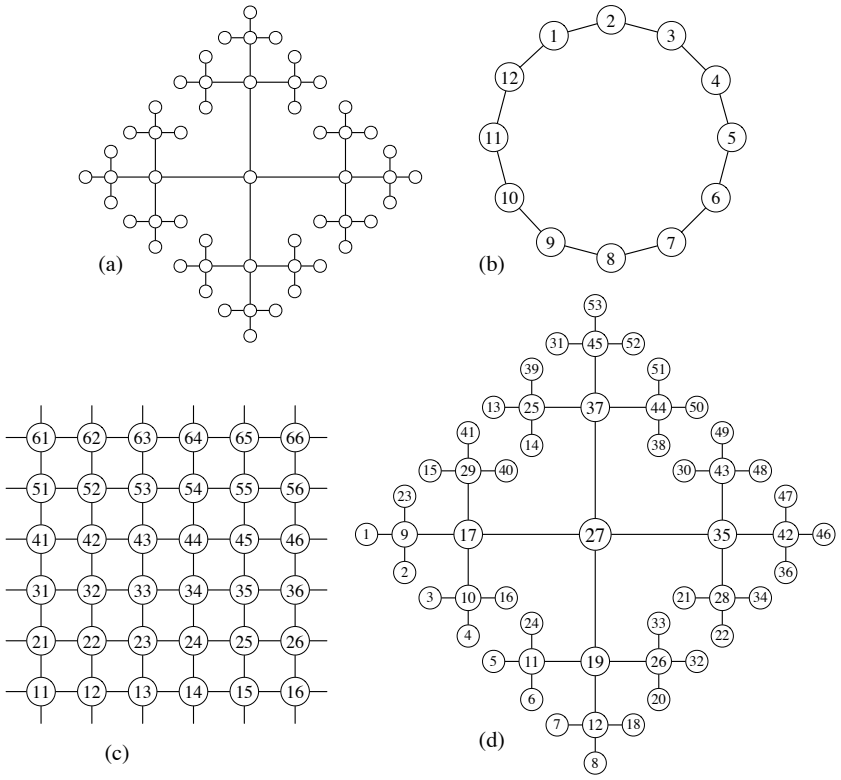


Figure 6: Examples of graphs that satisfy some of the desirable properties from Section 5.9.

However, what we can do is to put these ingredients together to prove a lower bound for maximal *fractional* matchings. We can show that maximal fractional matchings cannot be solved in time $o(\Delta)$ independently of n in the LOCAL model—recall that there is a matching upper bound of $O(\Delta)$. There are many technical details to be taken care of, but the key steps are as follows [14]:

- Prove a lower bound for the EC model, with the help of the unfold-and-mix technique (Section 5.6).
- EC $\approx$ PO: Use the properties of maximal fractional matchings to show that a lower bound for EC implies a lower bound for PO.
- PO $\approx$ OI: Use homogeneous graphs (Section 5.9) to show that a lower bound for PO implies a lower bound for OI.
- OI $\approx$ LOCAL: Use Ramsey's theorem to show that a lower bound for OI implies a lower bound for LOCAL.

Finally, we can prove that in the LOCAL model, randomised $f(\Delta)$-time algorithms cannot do any better than deterministic algorithms with this kind of graph problems, and we have the theorem that we have been looking for: even if we use randomised algorithms, and even if we can exploit the full power of the LOCAL model, there is no distributed algorithm that finds a maximal fractional matching in time $o(\Delta)$ independently of n.

6 Conclusions

So far we have identified the first pure coordination problem—maximal fractional matchings—with the following properties: the problem can be solved in time $O(\Delta)$ independently of n, and there is a matching lower bound showing that it cannot be solved in time $o(\Delta)$ independently of n.

Intuitively, the unfold-and-mix technique shows that there is need for local coordination with nodes that are at distance up to $\Theta(\Delta)$. All other parts of the proof—e.g. homogeneous graphs and Ramsey's theorem—are just technicalities that are needed to show that algorithms cannot "cheat" somehow by exploiting unique node identifiers and randomness.

6.1 Current Obstacles

Our hope is that we could prove that many other problems—for example, maximal matchings—also have a similar source of hardness that is related to local coordination with nodes at distance up to $\Theta(\Delta)$. We conjecture that, for example,

maximal matching cannot be solved in time $o(\Delta) + O(\log^* n)$ with any algorithm. In essence, we believe that the Panconesi–Rizzi algorithm [31] with a running time of $O(\Delta + \log^* n)$ is optimal for $n \gg \Delta$.

Currently, there seem to be two obstacles that prevent us from proving such a theorem, both of which are related to the term $\Theta(\log^* n)$ in the lower bound that we are looking for.

1. The final step OI $\approx$ LOCAL (Section 5.8): The Ramsey-based argument that we use to show that OI and LOCAL are equally strong fails for $\Theta(\log^* n)$-time algorithms.

2. The starting point in the EC model (Section 5.6): In time $\Theta(\log^* n)$ an algorithm can find a vertex colouring that breaks the symmetry between adjacent nodes. Therefore it is no longer possible to use self-loops to construct instances that are "fragile".

However, once again we can try to deal with these two obstacles one by one. It turns out that there is a natural graph problem that would let us focus on the second obstacle first—the problem is bipartite maximal matching that we already discussed in Section 3.5.

6.2 Roadmap for the Future

Recall that maximal matching in bipartite 2-coloured graphs can be solved in time $O(\Delta)$ independently of n. We conjecture that bipartite maximal matchings are a pure coordination problem that cannot be solved in time $o(\Delta)$ independently of n. The current techniques are not yet sufficient to prove it, but it seems that we are now facing just one obstacle—how to use the unfold-and-mix technique to construct "fragile" instances even in the presence of an edge colouring that breaks symmetry.

This suggests the following roadmap for future research:

1. Extend the unfold-and-mix technique so that we can prove a linear-in-Δ bound for bipartite maximal matchings.

2. Then extend the Ramsey-based argument so that we can prove a similar bound for maximal matchings in general.

3. Then extend the techniques so that we can prove similar bounds for other coordination problems, for example, independent sets, vertex colourings, and edge colourings.

This seems to be a long road ahead, but it could lead to a resolution of major open questions related to distributed time complexity. Such results could find applications also in other areas of theoretical computer science. In prior work, tight lower bounds for distributed symmetry breaking have implied tight lower bounds for e.g. decision tree complexity and models of parallel computing [11, 30], and perhaps tight lower bounds for local coordination would find similar applications.

Acknowledgements

This article is based on the material that I presented in the ADGA 2014 workshop, http://adga2014.hiit.fi/. Many thanks to Christoph Lenzen for inviting me to give the talk, to the workshop participants for discussions, to Stefan Schmid for asking me to write this article and for his feedback on it, to Przemysław Uznański and Tuomo Lempiäinen for their helpful comments, and to my coauthors Mika Göös and Juho Hirvonen without whom the results described in this article would not even exist.

References

[1] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. doi:10.1016/0196-6774(86)90019-2.

[2] Matti Åstrand, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. Local algorithms in (weakly) coloured graphs, 2010. arXiv:1002.0125.

[3] Matti Åstrand and Jukka Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proc. 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2010)*, pages 294–302. ACM Press, 2010. doi:10.1145/1810479.1810533.

[4] Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981. doi:10.1016/0196-6774(81)90020-1.

[5] Leonid Barenboim and Michael Elkin. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC 2009)*, pages 111–120. ACM Press, 2009. doi:10.1145/1536414.1536432.

[6] Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool, 2013. doi:10.2200/S00520ED1V01Y201307DCT011.

[7] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. In *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 321–330. IEEE Computer Society Press, 2012. doi:10.1109/FOCS.2012.60.

[8] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986. doi:10.1016/S0019-9958(86)80023-7.

[9] Andrzej Czygrinow, Michał Hańćkowiak, Edyta Szymańska, and Wojciech Wawrzyniak. Distributed 2-approximation algorithm for the semi-matching problem. In *Proc. 26th International Symposium on Distributed Computing (DISC 2012)*, volume 7611 of *Lecture Notes in Computer Science*, pages 210–222. Springer, 2012. doi:10.1007/978-3-642-33651-5_15.

[10] Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Proc. 22nd International Symposium on Distributed Computing (DISC 2008)*, volume 5218 of *Lecture Notes in Computer Science*, pages 78–92. Springer, 2008. doi:10.1007/978-3-540-87779-0_6.

[11] Faith E. Fich and Vijaya Ramachandran. Lower bounds for parallel computation on linked structures. In *Proc. 2nd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 1990)*, pages 109–116. ACM Press, 1990. doi:10.1145/97444.97676.

[12] Alex Gamburd, Shlomo Hoory, Mehrdad Shahshahani, Aner Shalev, and Balint Virág. On the girth of random Cayley graphs. *Random Structures & Algorithms*, 35(1):100–117, 2009. doi:10.1002/rsa.20266.

[13] Mika Göös, Juho Hirvonen, and Jukka Suomela. Lower bounds for local approximation. *Journal of the ACM*, 60(5):39:1–23, 2013. doi:10.1145/2528405. arXiv:1201.6675.

[14] Mika Göös, Juho Hirvonen, and Jukka Suomela. Linear-in-Δ lower bounds in the LOCAL model. In *Proc. 33rd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2014)*, pages 86–95. ACM Press, 2014. doi:10.1145/2611462.2611467. arXiv:1304.1007.

[15] Mika Göös and Jukka Suomela. No sublogarithmic-time approximation scheme for bipartite vertex cover. *Distributed Computing*, 27(6):435–443, 2014. doi:10.1007/s00446-013-0194-z. arXiv:1205.4605.

[16] Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey Theory*. John Wiley & Sons, New York, 1980.

[17] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998)*, pages 219–225. Society for Industrial and Applied Mathematics, 1998.

[18] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. *SIAM Journal on Discrete Mathematics*, 15(1):41–57, 2001. doi:10.1137/S0895480100373121.

[19] Juho Hirvonen and Jukka Suomela. Distributed maximal matching: greedy is optimal. In *Proc. 31st Annual ACM Symposium on Principles of Distributed Computing (PODC 2012)*, pages 165–174. ACM Press, 2012. doi:10.1145/2332432.2332464. arXiv:1110.0367.

[20] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):77–80, 1986. doi:10.1016/0020-0190(86)90144-4.

[21] Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *Proc. 21st Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2009)*, pages 138–144. ACM Press, 2009. doi:10.1145/1583991.1584032.

[22] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 300–309. ACM Press, 2004. doi:10.1145/1011767.1011811.

[23] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, pages 980–989. ACM Press, 2006. doi:10.1145/1109557.1109666.

[24] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: lower and upper bounds, 2010. arXiv:1011.5470.

[25] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

[26] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. doi:10.1137/0215074.

[27] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Francisco, 1996.

[28] Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991. doi:10.1137/0404036.

[29] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

[30] Noam Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. doi:10.1137/0220062.

[31] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed Computing*, 14(2):97–100, 2001. doi:10.1007/PL00008932.

[32] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[33] Jukka Suomela. Survey of local algorithms. *ACM Computing Surveys*, 45(2):24:1–40, 2013. doi:10.1145/2431211.2431223. http://www.cs.helsinki.fi/local-survey/.

[34] Jukka Suomela. *Distributed Algorithms*. 2014. Online textbook. http://users.ics.aalto.fi/suomela/da/.

The Education Column

BY

Juraj Hromkovič

Department of Computer Science
ETH Zürich
Universitätstrasse 6, 8092 Zürich, Switzerland
juraj.hromkovic@inf.ethz.ch

BEATCS no 115

HOMO INFORMATICUS

WHY COMPUTER SCIENCE FUNDAMENTALS ARE AN UNAVOIDABLE PART OF HUMAN CULTURE AND HOW TO TEACH THEM

Juraj Hromkovič
Department of Computer Science
ETH Zürich
juraj.hromkovic@inf.ethz.ch

Abstract

The goal of this article is on one hand to introduce informatics as a scientific discipline in the general context of science and to outline its relationships especially to mathematics and engineering, and on the other hand to propose a way how to integrate computer science into school education.

1 Introduction

The development of human society is mainly determined by the ability to derive knowledge and to find efficient ways of applying it. Let us explain this claim more carefully. Human beings sample experiences by observations and experiments and use them to generate knowledge by combining their experience with logical thinking. The derived knowledge is used to develop procedures in order to reach concrete goals. A crucial point in our considerations is that to use such procedures one usually does not need to fully understand the knowledge used to obtain such procedures. To an even higher extent, one does not need to understand the way in which this knowledge was derived and verified. Let us illustrate this on a simple example. The famous theorem of Pythagoras claims $c^2 = a^2 + b^2$ in any right triangle, where c is the length of the longest side (hypotenuse). This theorem was used to fix the right angles when building houses and temples in the classical antiquity. The workers only needed to build a triangle of sizes 5 units, 4 units, and 3 units in order to get a right angle, and for sure they did not need to understand how the theorem of Pythagoras was discovered and how it was proven. Hence, to successfully apply the derived knowledge, one did not need to master the high

qualification of an investigator. In this way, scientists changed and still change society and became a crucial factor in human development.

Computer science was created as a natural step forward in the above described development, when the following two conditions were satisfied.

1. Using the exact language of mathematics, one was able to discover and describe procedures in such a way that no intellect was needed to apply them. Executing them step by step, everything was unambiguous, and no educated or trained person was needed to find the right interpretation of the particular instructions of the procedure executed.

2. The technology enabling to execute the discovered procedures by machines was developed, and languages providing the opportunity to "explain" universal machines what they have to do were designed.

In this context, we speak about automation. We let the machine execute not only physical work, but also such human work considered as intellectual work in the past. This is the reason why computer science is a mixture of mathematics and engineering. On the one hand, one uses the concepts, methods, and the language of mathematics in order to understand the related things so exactly that algorithms as unambiguously interpretable procedures can be developed in order to solve a variety of problems everywhere in science, technology, and everyday life. Here, mathematics is the instrument that has to be mastered together with the specific area in which one tries to solve problems. On the other hand, developing and improving the enabling technology is mostly engineering. This is not only about hardware, but especially about software enabling us to communicate with computers conveniently in high-level languages.

We see that computer science is a part of the natural development of science, and it became a discipline that is crucial for the performance of the human society. This contextual view is important when thinking about what computer science actual is, and how to teach computer science in schools. We are asked not to teach specific isolated concepts, but the discoveries, i. e., the paths from the motivation coming from the general context of science to the final products of the research work and engineering. As in its fundamentals computer science is very strongly related to mathematics as its basic research instrument, we start with the question "What is mathematics and how to teach it?" in the next section, and use the derived point of view in the final section to propose the way of teaching computer science in schools. Before the final section, however, we discuss engineering as a missing subject in schools in Section 3. Again, we use this discussion in order to show in the final section how teaching computer science can contribute to understanding some basic concepts of engineering and how it integrates them in the school curriculum.

2 Mathematics as the Language of Science and Consequences of this View for Teaching it

What is mathematics? If you pose this question, you can get a lot of different answers; frequently, even the response that this question is too hard to be answered satisfactorily. A mathematician or a university student can tell you with high probability that she or he proves theorems and thus investigates the structure and the properties of artificial mathematical objects, which can be useful to model reality, or the relationships between different such objects. A high-school student can tell you that mathematics consists of calculations that can be used to solve some classes of mathematical tasks (also called problems) such as solving quadratic equations or systems of linear equations or analyzing the properties of a curve. Obviously, you can also get a response that mathematics is something that is hard to understand, and that every "normal" human being can exist and be successful without it. This is a frequent answer, and unfortunately it is more of a rule than an exception.

Mathematics is the most powerful instrument humans ever have developed in order to investigate the world around us. But it is taught in such a way that the students do not realize this fact. Especially in high schools, they learn methods (algorithms) to solve some given problems (for instance, to find a maximum of a function). This may be also viewed positively as an intellectual challenge, because several methods are not easy to manage. But the bad news is that they can learn to successfully apply methods for solving different problems without really understanding why these methods work properly. High-school students often do not have a good intuition of what infinity or limits are about, but they use these concepts in a fuzzy way in order to analyze artificial functions without any relation to reality. What are we doing wrong? A lot. We have to start to think first about what mathematics is, and then to try to find a way how to teach mathematics in a proper way.

From my point of view, the best way to view mathematics is as a special language developed for science, i. e., for knowledge generation. Going a few thousand years back, people wanted to discover "objective" knowledge. The important word here is the word objective. If you want to reach that, first of all you need a language in which each statement has an unambiguous interpretation for everybody who mastered this language. How to reach this? First of all, you need to give an absolutely exact meaning to the words (notions) you use, because the words are semantically the corner stones of each language. In this context, mathematicians speak about axioms. Many people have the wrong impression that axioms are claims in whose truthfulness we believe, but of which one is not able to prove that they are true. This is wrong. Axioms are the precise definitions of basic no-

tions that describe our intuition about the meaning of these notions. Probably the first concepts people tried to fix were notions such as number, equality, infinity, point, line, distance, etc. What is very important to observe is that people needed hundreds and in some cases thousands of years to come up with such definitions that the community of philosophers and later mathematicians has accepted. Why was all of this done? First of all, the language of mathematics is able to describe objects, structures, properties, and relationships in an unambiguous way. In this context we speak about the "descriptional" power of mathematics. Thus, people were able to formulate exact claims this way, and so to express our knowledge unambiguously. But this was only one side of the language of mathematics. The language of mathematics was also used to derive new knowledge from the given knowledge, i. e., as a knowledge generator. Leibniz formulated this role of mathematics in a very nice way. He wanted to omit all political discussions and fighting in different committees by simply expressing the real problems in the language of mathematics, and then using calculations and logical derivations to obtain the right solution. Interestingly, he called this "automation" of the human work. By now we know that this dream of Leibniz cannot be achieved. There are two reasons for that. First of all, because of its exactness, the language of mathematics is restricted in its descriptional power, and so we cannot translate all real-world problems into this language. Secondly, one of the most important discoveries of the last century made by GÎŞÂűdel tells us that the argumentational power of the language of mathematics is smaller than its descriptional power. This means that one can formulate claims in current mathematics for which there do not exist proofs of whether they are true or not.

What do scientists learn from that? The development of the language of mathematics is similar to the development of natural languages. You need to create new words and describe their meaning in order to increase its descriptional power, and to be able to speak about things you were not able to speak about before. Moreover, you need new concepts and words to be able to argue about matters you were not able to argue before, i. e., in order to strengthen your capability of deriving new knowledge by thinking. A very important point is that the process of developing the language of mathematics is infinite. As long the number of basic, axiomatic words of the language of mathematics is finite, one can always formulate claims that are not provable within this mathematical language, and one has to introduce new words in order to be able to prove them and so to make new discoveries. A very nice example is the notion of infinity. Nobody saw anything infinite in the real world, and even the physicists believe that the universe is finite. This means that infinity is something artificial, simply an artificial product of mathematics. But without this concept, most of the current science would not exist. Without the notion of infinity, there would be no concept of limits and so we would not be able to exactly express notions like actual speed or actual acceleration. Our

science simply would be somewhere before the discoveries of Newton. Hence, without the "artificial" concept of infinity, one is strongly restricted in the ability to discover our finite, real world.

Another example of a crucial notion is the concept of probability. Most of the sciences, even the non-exact ones such as didactics, psychology, medicine, and economy, heavily use this concept to model and investigate reality, and if they would make predictions without this concept, they would have serious trouble to convince society that their results are trustable to some extent and not only expert opinions.

Why did we focus on the view of mathematics presented above? Because it is the nature of mathematics that shows us which changes are necessary in order to improve the education in mathematics for everybody. Based on this, we recommend to adopt the following concepts.

1. Focus more on the genesis of the fundamental notions (concepts) of mathematics. To define them took centuries, to prove most of the theorems took a few years. Each new concept enabled to investigate so many things that no single discovery can compete with introducing a fundamental concept. The strengthening of mathematics as a research instrument is the main job of mathematics, and deriving new concepts (not necessarily on the axiomatic level) in mathematics provides the best, true picture of its nature. Without this, nobody can really understand its role and usefulness. And only if one understands the genesis of mathematics as the development of a language of science and as a research instrument, one can be able to apply it regularly to all areas of our life. Teaching mathematics this way can completely change the behaviour of the members of our society. Instead of memorizing and sampling facts provided, one would start to verify the degree of trustability of claims sold as knowledge and to understand to which extent and under which conditions one is allowed to take them seriously.

2. Concrete examples first, abstraction as a final discovery. One first has to touch concrete problem instances and objects in order to get some intuition about their properties. Then, one can formalize her or his intuition into a formal concept. One has to follow the natural way of discovering that usually goes from concrete to abstract. To sell methods and theorems as final products is as poor as teaching manuals for washing machines or Microsoft office instead of teaching discoveries of physics, mechanics, and computer science that enabled to develop these products.

3. Teach algorithmics instead of training calculation methods. Pupils learn in schools to multiply arbitrarily large integers in decimal representations, to solve quadratic equations and systems of linear equations, or to analyze

functions, etc. In all cases, most pupils learn to apply given methods, but most of them do not understand why they work. It is more of a challenging memorizing than a deep understanding of the nature of the algorithms used. One has to start to introduce problems instead of presenting methods solving them, and ask the pupils and students to solve concrete problem instances first and finally to discover an algorithm as a robust procedure that is able to solve any of the infinitely many concrete problem instances of the given problem. Discovering algorithms as well-functioning calculation procedures offers another quality of education in mathematics than executing a given calculation method, which any pocket calculator can do faster and more reliable. Teach programming as the art of exactly describing the methods discovered in an unambiguously interpretable way in the language of the machines, and strengthen the ability of exact communication this way.

4. Teach the principles of correct argumentation. Teach the notions of implication and quantifiers, and train direct and indirect proofs. Do not believe that the pupils in high schools cannot learn to verify and to derive simple proofs. They did not manage this in the past, because there was no effort made to teach proving claims, or most effort in this direction was done in a wrong way.

5. Guarantee the opportunity to the pupils to deal with the subjects as many time as needed at an individual speed. Mathematics is one of the sciences that needs a large number of iterative touchings of particular topics until one is allowed to say "Eureka," and gets a reasonable understanding of what it is about. The trouble is that no teacher can assure this by herself or himself for everybody in the class. Another problem is that most textbooks of mathematics are good collections of exercises, but explanations are written more for teachers than for pupils. One way out is to change the style of the textbooks. The textbooks should be written in such a way that pupils and students would be able to learn from them by their own with a minimal support from outside. Partitioning the discoveries into a number of small, natural steps written in the language mastered by the pupils at the corresponding age, and regularly giving the opportunity to verify whether one understands the topic up to now properly are some of the basic principles used to create good textbooks for mathematics.

Finally, one can ask how to reach the new teaching style for mathematics described above. For sure, one cannot ask the high-school teachers to make this change without showing them how to do this in detail. One also cannot ask educationalists, who do not have a sufficiently deep contextual knowledge of mathematics to master these changes. The movement has to start at the universities,

where the teaching style has to change first. In order to speed up this process in Switzerland, in our department at ETH ZÎŞÎŇrich we develop new textbooks for teaching different topics of mathematics and computer science for all school levels. Our experiments prove that mathematics can become one of the most favorite subjects of pupils and students if taught in the way described above. Students can successfully master topics that were considered to be too hard for them before, and the marks in mathematics can be significantly higher than the average marks over all topics.

For me, it is not a question of whether the proposed evolution of the education in mathematics will come. It is only the question of the time at which particular countries will need to adopt it. Since this is a service for the future generation, the earlier the better.

3 Why Engineering is Not Allowed Not to be a Part of Basic Education

As mentioned in the introduction, human society uses the knowledge discovered in order to reach different goals more efficiently by developing various procedures or different products. This is highly creative work that is beyond the pure learning of facts and making calculations that can be completely automatized. The whole process of engineering work starts with a description of the goals to be achieved. After that, one starts to combine experience and fundamental knowledge of science in order to design a solution that has to be implemented as a prototype. Next, one has to test this prototype, modify, and improve it until an acceptable product is produced.

In today's schools, we find almost nothing about the concept of iterative specifying, testing, modifying, and improving the product of our work, not speaking about fundamental constructive ways of creating original products. But this is fundamental to human activities since forever. The current school systems ignore this fact to a high extent and are more about teaching to memorize than teaching to work in a creative way. One can explain this educational misconception by the fact that, in contrast to basic scientific models of reality, the engineering work is heavily dependent on experience that can be hardly formalized and thus taught. The work of engineers as human experts cannot be described by an algorithm (a method). However, this must not be a reason to remove engineering from education, because one has also to learn to build her or his own experience over a longer period of time in order to become an expert for a special area.

The crucial fact we want to point out is that computer science mastered to formalize several basic concepts of engineering and made them available to our

schools as a result. Teaching computer science is a chance to introduce engineering as a highly creative, constructive activity to our educational system.

4 Teaching Computer Science as a Fundamental Step in the Evolution of Our Educational Systems

In the two previous sections, we already outlined, with respect to improving teaching of mathematics and introducing engineering, the principal contributions that could be offered by teaching computer science in schools in a proper way. This word "proper" is crucial for us, and thus we start by listing what we are not allowed to do if we want to avoid a disaster when introducing computer science to schools. In what follows, we present the most frequent mistakes that already caused frustrations in different countries.

1. To teach how to work with concrete software products and call it computer science. This activity destroyed the image of computer science as a scientific discipline in the past.

2. To let computer science be taught as a part of "social media" by teachers educated in human sciences only, and focusing on social, emotional and psychological aspects of communication by new technologies.

3. To choose the topic to be taught by committees of experts offering their favourite topic without looking at the whole context of science as presented above in Section 2.

4. To sell computer science as the ability to work with computers.

5. To sell computer science as a special branch of mathematics.

6. To sell computer science as a pure engineering discipline.

7. To try to teach the newest achievements of computer science. Think about what would happen if physics would try to do that instead of following the history, and thus developing step by step our view of the physical world.

8. To try to sell computer science as a joy much easier than mathematics and physics by avoiding any depth and thus a spiral curriculum, and instead presenting one simple application after the other.

9. Going too much into technical details about concrete programming languages, software systems, or hardware.

While 1, 2, and 4 have been the main reasons for destroying the image of computer science in the society, currently the points 7 and 8 are the major danger for establishing computer science as a school subject.

After listing what we are not allowed to do, it is now time to switch to positive recommendations and conceptual work. We do not want to make a proposal for the content of a computer science curriculum, because our goal is not to go too much into detail, and because, for sure, there are various good implementations of the computer science subject in schools. What we try here is to recommend a strategy and principles that are useful for designing a computer science curriculum that can be accepted as a fundamental part of education in its generality for everybody, and that essentially contributes to...

1. the understanding of our world (in this case with the focus on the artificial world created by humans),

2. developing our way of thinking in a dimension that cannot be compensated by teaching other school subjects,

3. providing knowledge that is useful and sometimes even expected for the study of a variety of specialized scientific disciplines later (university studies, etc.).

As a byproduct, we have to aim to improve teaching overall, especially by strengthening the subjects mathematics and natural sciences. We already presented the basic strategy how to design a computer science curriculum in Section 2 about mathematics. We have to follow the genesis of computer science and think about motivations and fundamental concepts introduced and discovered by computer scientists from the point of view of science as a whole. For sure, we have to think about or even discover which concepts are available to which extent in which age, and to follow all the ideas for creating good teaching materials as presented in Section 2. Let us be more concrete and present a few examples.

One can decide to introduce programming at the age between 8 and 14. The first step is to deal with abstractions that enable us to unambiguously describe the problem instances. Then we teach to sample experience by trying to find solutions to concrete, special problem instances, whose size and complexity may grow with growing experience. After some time, one can develop a strategy that works successfully for a small collection of problem instances that we subsequently call a problem. Having a solution strategy, one has to learn to communicate it, i. e., to unambiguously describe it for anybody else. After that, we are allowed to start teaching proper programming by describing our strategy as a program in a suitable programming language. We are not allowed to teach the list of all instructions (fundamental words) of a programming language. We have to start with very

few (10 to 15) fundamental instructions, and use modularity to create new words (instructions) in order to make our communication with the computer more convenient. After writing programs, we let them run in order to verify their correct functionality and learn to correct, improve, and modify our programs in order to get a final product with which we are satisfied. Let us list some of the added values when teaching the introduction to programming in the way described above.

1. Training and strengthening the abstraction in representing real situations by drawing graphs, writing lists or tables with different kinds of elements.

2. Contributing to teaching mathematics by searching for a solving strategy, instead of simply learning a method as a given product of the work of others.

3. Strengthening the ability to express matters and procedures in an exact way, and so to improve the way used to communicate.

4. Recognizing that a language is not a given final product of human work, but that any language is continuously developed, and that in case of a programming language one can develop the language on her or his own with respect to her or his personal demand.

5. Defining new instructions by describing the meaning of the new words by subprograms, one learns the principle of a modular design that is common and fundamental in engineering.

6. Introducing the concepts of testing, verifying, modifying, and improving is the first contact with the creative, constructive work of engineers.

A really good teaching sequence for introductory programming can be created if one focuses on the above listed added values and not on technical details of programming languages and other software used.

Another nice example is teaching cryptography. Cryptography can be viewed as the history of developing the notion "secure cryptosystem." One can start with the historical examples in order to introduce the basic terms decryption, encryption, key, and cryptosystem with a lot of creative work by designing and breaking new, own cryptosystems. After defining the concept of "security" by Kerckhoff, one can build the bridge to probability theory. The concept of probability was used to design new cryptosystems and later to break them. One can wonderfully understand the importance and the usefulness of the concept of probability studying the history of secret communication in this way. Then one can introduce the formal mathematical definition of absolutely secure cryptosystems with respect to the concept of probability, and recognize that such system cannot be built

for practical purposes. Finally, the concept of computational complexity offering public-key cryptosystems is the way out, leading to the recent e-commerce.

What we try to repeatedly present as the key strategy is to follow the history of the discoveries of particular concepts, methods, and ideas, and not to try to sell finalized products of science. The creative work is the most (and may be even the only really) exciting part of the study. Let us teach creativity by repeatedly discovering things that were already discovered, up to the point where one is able to discover something completely new. Forget about teaching facts, teach how to verify the trustability of claims made by others. We are lucky, because we are allowed to create a curriculum for a completely new subject. We can implement principles, which the other subjects still did not recognize, and so contribute to the evolution of the system of education. For those who would like to see detailed implementations of the design principles presented above, we recommend to following textbooks from our production [1, 2, 3, 6] or the book "Algorithmic Adventures – from Knowledge to Magic" [4, 5].

References

[1] H.-J. Böckenhauer and J. Hromkovič. *Formale Sprachen.* Springer Vieweg, 2013.

[2] K. Freiermuth, J. Hromkovič, L. Keller, and B. Steffen. *Einführung in die Kryptologie.* 2nd Edition. Springer Vieweg, 2014.

[3] J. Hromkovič. *Berechenbarkeit.* Vieweg+Teubner, 2011.

[4] J. Hromkovič. *Sieben Wunder der Informatik – Eine Reise an die Grenze des Machbaren.* Vieweg+Teubner, 2008.

[5] J. Hromkovič. *Algorithmic Adventures – From Knowledge to Magic.* Springer, 2009.

[6] J. Hromkovič. *Einführung in die Programmierung mit LOGO.* 3rd Edition. Springer Vieweg, 2014.

THE LOGIC IN COMPUTER SCIENCE COLUMN

BY

YURI GUREVICH

Microsoft Research
One Microsoft Way, Redmond WA 98052, USA
gurevich@microsoft.com

BEATCS no 115

NEGATIVE PROBABILITY

Andreas Blass
Mathematics Department, University of Michigan
Ann Arbor, MI 48109, USA, ablass@umich.edu

Yuri Gurevich

Abstract

The uncertainty principle asserts a limit to the precision with which position x and momentum p of a particle can be known simultaneously. You may know the probability distributions of x and p individually but the joint distribution makes no physical sense. Yet Wigner exhibited such a joint distribution $f(x, p)$. There was, however, a little trouble with it: some of its values were negative. Nevertheless Wigner's discovery attracted attention and found applications. There are other joint distribution, all with negative values, which produce the correct marginal distributions of x and p. But only Wigner's distribution produces the correct marginal distributions for all linear combinations of position and momentum. We offer a simple proof of the uniqueness and discuss related issues.

1 Introduction

"Trying to think of negative probabilities," wrote Richard Feynman, "gave me a cultural shock at first" [6]. Yet quantum physicists tolerate negative probabilities. Feynman himself studied, in the cited paper, a probabilistic trial with four outcomes with probabilities 0.6, −0.1, 0.3 and 0.2.

We were puzzled. The standard interpretation of probabilities defines the probability of an event as the limit of its relative frequency in a large number of trials. "The mathematical theory of probability gains practical value and an intuitive meaning in connection with real or conceptual experiments" [5, §I.1]. Negative probabilities are obviously inconsistent with the frequentist interpretation. Of course, that interpretation comes with a tacit assumption that every outcome is observable. In quantum physics some outcomes may be unobservable. This weakens the frequentist argument against negative probabilities but does not shed much light on the meaning of negative probabilities.

In the discrete case, a probabilistic trial can be given just by a set of outcomes and a probability function that assigns nonnegative reals to outcomes. One can generalize the notion of probabilistic trial by allowing negative values of the probabilistic function. Feynman draws an analogy between this generalization and the generalization from positive numbers, say of apples, to integers. But a negative number of apples may be naturally interpreted as the number of apples owed to other parties. We don't know any remotely natural interpretation of negative probabilities.

We attempted to have a closer look on what goes on.

Heisenberg's uncertainty principle asserts a limit to the precision with which position x and momentum p of a particle can be known simultaneously: $\sigma_x\sigma_p \geq \hbar/2$ where σ_x, σ_p are the standard deviations and $\hbar$ is the (reduced) Planck constant. You may know the probability distributions (or the density function or the probability function; we will use the three terms as synonyms) of x and p individually but the joint probability distribution with these marginal distributions of x and p makes no physical sense[1]. Does it make mathematical sense? More exactly, does there exist a joint distribution with the given marginal distributions of x and p.

In 1932, Eugene Wigner exhibited such a joint distribution [20]. There was, however, a little trouble with Wigner's function. Some of its values were negative. The function, Wigner admits, "cannot be really interpreted as the simultaneous probability for coordinates and momenta." But this, he continues, "must not hinder the use of it in calculations as an auxiliary function which obeys many relations we would expect from such a probability" [20]. Probabilistic functions that can take negative values became known as *quasi-probability distributions*.

Richard Feynman described a specific quasi-probability distribution for discrete quantities, two components of a particle's spin [6]. The uncertainty principle implies that these two quantities can't have definite values simultaneously. So it seems plausible that an attempt to assign joint probabilities would again, as in Wigner's case, lead to something strange — like negative probabilities. "Trying to think of negative probabilities gave me a cultural shock at first ... It is usual to suppose that, since the probabilities of events must be positive, a theory which gives negative numbers for such quantities must be absurd. I should show here how negative probabilities might be interpreted" [6]. His attitude toward negative probabilities echoes that of Wigner: a quasi-probability distribution may be used to simplify intermediate computations. The meaning of negative probabilities remains unclear. Those intermediate computations may not have any physical sense. But if the final results make physical sense and can be tested then the use

[1]For general information about joint probability distributions and their marginal distributions see [5, §IX.1]

of a quasi-probability is justified.

It bothered us that both Wigner and Feynman apparently pull their quasi-probability distributions from thin air. In particular, Wigner writes that his function "was found by L. Szilárd and the present author some years ago for another purpose" [20], but he doesn't give a reference, and he doesn't give even a hint about what that other purpose was. He also says that there are lots of other functions that would serve as well, but none without negative values. He adds that his function "seemed the simplest."

We investigated the matter and made some progress. We found a characterization of Wigner's function that might be considered objective.

Proposition 1 (Wigner Uniqueness)**.** *Wigner's function is the unique quasi-distribution on the phase space that yields the correct marginal distributions not only for position and momentum but for all their linear combinations.*

> **Quisani**[2]**:** Wait, I don't understand the proposition. Wigner's function is not a true distribution, so the notion of marginal distribution of Wigner's function isn't defined. Also, what does it mean for a marginal to be correct?
>
> **Authors**[3]**:** The standard definition of marginals works also for quasi-probability distributions. A marginal distribution is correct if it coincides with the prediction of quantum mechanics. We'll return to these issues in §3 and §4 respectively.
>
> **Q:** To form a linear combination $ax+bp$ of position x and momentum p you add x, which has units of length like centimeters, and p, which has momentum units like gram centimeters per second. That makes no sense.
>
> **A:** We are adding ax and bp. Take a to be a momentum and b to be a length; then both ax and bp have units of action (like gram centimeters squared per second), so they can be added. If you want to make $ax + bp$ a pure number, divide by $\hbar$.
>
> **Q:** Finally, is it obvious that Wigner's quasi-distribution is not determined already by the correct marginal distributions for just the position and momentum, without taking into account other linear combinations?
>
> **A:** This is obvious. There are modifications of Wigner's quasi-distribution that still give the correct marginal distributions for position and momentum. For an easy example, choose a rectangle R

[2]Readers of this column may remember Quisani, an inquisitive former student of the second author.

[3]speaking one at a time

> centered at the origin in the (x, p) phase plane and modify Wigner's $f(x, p)$ by adding a constant c (resp. subtracting c) when $(x, p) \in R$ and the signs of x and p are the same (resp. different). For a smoother modification, you could add $cxp \exp(-ax^2 - bp^2)$ where a, b, c are positive constants (of appropriate dimensions).

The idea to consider linear combinations of position and momentum came from Wigner's paper [20] where he mentions that projections to such linear combinations preserve the expectations. In fact, the projections give rise to the correct marginals. This led us to the proposition.

In the case of Feynman's quasi-distribution mentioned above, one can't use linear combinations of those two spin components to characterize the distribution. Nor is there a characterization using the spin component in yet another direction. Furthermore, if we only require the correct marginal distributions for the x and z spins, then there are genuine, nonnegative joint probability distributions with those marginals.

Our investigation was supplemented by digging into the literature and talking to our colleagues, especially Nathan Wiebe. That brought us to "Quantum Mechanics in Phase Spaces: An Overview with Selected Papers" [22]. It turned out that there was another, much earlier, approach to characterizing the Wigner quasi-probability distribution. The main ingredient for that earlier approach is a proposal by Hermann Weyl [19, §IV.14] for associating Hermitian operators on L^2 to well-behaved functions $g(x, p)$ of position and momentum. José Enrique Moyal used Weyl's correspondence to characterize Wigner's quasi-distribution in terms of only the expectation values but for a wider class of functions rather than the marginal distributions for just the linear functions of position and momentum [10]. There is a trade-off here. The class of functions is wider but the feature to match is narrower.

George Baker proved that any quasi-distribution on the position-momentum phase-space, satisfying his "quasi-probability distributional formulation of quantum mechanics," is the Wigner function [2]. The problem of an objective characterization of Wigner's function also attracted the attention of Wigner himself [21, 12].

The volume [22] does not contain "our" characterization of Wigner's function but it is known and due to Jacqueline and Pierre Bertrand [3]. They found an astute name for the approach: tomographic. The tomographic approach gives an additional confirmation of the fact that behavior of our quantum system is not classical. The approach can be used to establish that the behavior of some other quantum systems is not classical. It has indeed been used that way in quantum optics; see [15, 13] for example.

Still, in our judgment, our proof of the uniqueness of Wigner's function is

simpler and more direct than any other in the literature, and so we present it in §4. In section §5 we establish Moyal's characterization of Wigner's function.

§6 contains a cursory discussion related to Feynman's four-outcome quasi-distribution. All our observations on that issue happened to be known as well, but we have yet to research the history of the negative probabilities in the discrete case. We intend to address the discrete case elsewhere.

> **Q:** So what is the meaning of negative probability?
>
> **A:** We don't know.
>
> **Q:** The use of negative probabilities to validate the quantum character of a quantum system reminds me proofs by contradiction. Assume that the behavior is classical, produce a unique joint distribution, prove the existence of negative values and establish a contradiction. If this is the only use of negative probabilities then there is no need to interpret them semantically.
>
> **A:** There are some attempts to use negative probabilities as a measure of "quantumness" [17, 18]. We think that the jury is still out.
>
> **Q:** I have yet another question. Recently, in this very column, Samson Abramsky wrote about contextuality which is another manifestation of non-classical behavior of quantum systems [1]. I wonder what is the relation, if any, between negative probabilities and contextuality.
>
> **A:** The discussion of that relation is beyond the scope of this paper. But please have a look at Robert Spekkens's article [16] with a rather telling title "Negativity and contextuality are equivalent notions of nonclassicality."

Acknowledgment

Many thanks to Nathan Wiebe who was our guide to the literature and the state of art on quasi-probabilities.

2 Preliminaries

We tried to make the paper as accessible as possible; hence this section. We still assume some familiarity with mathematical analysis. By default in this paper integrals are from $-\infty$ to $+\infty$. The baby quantum theory that we use is covered in §3 of book [8] titled "A first approach to quantum mechanics,".

2.1 Fourier transform

The forward Fourier transform sends a function $f(x)$ to

$$\hat{f}(\xi) = \frac{1}{\sqrt{2\pi}} \int f(x)\, e^{-i\xi x}\, dx.$$

and the inverse Fourier transform sends a function $g(\xi)$ to

$$\check{g}(x) = \frac{1}{\sqrt{2\pi}} \int g(\xi) e^{i\xi x}\, d\xi,$$

Mathematically x and ξ are real variables. In applications, the dimension of ξ is the inverse of that of x so that ξx is a pure number.

The forward and inverse Fourier transforms are defined also for functions of several variables. In particular,

$$\hat{f}(\xi, \eta) = \frac{1}{2\pi} \iint f(x, y)\, e^{-i(\xi x + \eta y)}\, dx\, dy,$$
$$\check{g}(x, y) = \frac{1}{2\pi} \iint g(\xi, \eta) e^{i(\xi x + \eta y)}\, d\xi\, d\eta.$$

> **Q:** What about the convergence of the integrals? Are you going to ignore such details?
>
> **A:** Yes, we are going to ignore such details. But Fourier transforms are used, with full mathematical rigor, even in some situations where the integrals don't converge.
>
> **Q:** I do not understand this.
>
> **A:** The idea is to first define the Fourier transform as an operator on nice functions in $L^2(\mathbb{R})$, for which the integrals clearly converge. Informally a function $f(x)$ is nice if it and its derivatives $f'(x), f''(x), f'''(x), \ldots$ approach zero very rapidly as $x \to \infty$. The Fourier transform is an isometry on these nice functions, and the nice functions are dense in $L^2(\mathbb{R})$, so the isometry extends to all of L^2. Details can be found in books on real analysis, like [9] and [14]; alternatively, see [8, Appendix A.3.2].

2.2 Dirac's delta function

Dirac's δ-function is a generalized function such that for any nice function f,

$$\int f(x)\delta(x)dx = f(0).$$

It follows that

$$\int f(x)\delta(x-a)dx = \int f(x+a)\delta(x)dx = f(a).$$

Some divergent integrals, e.g. $\int e^{itx}dt$, can be seen as generalized functions in that sense. In fact, as generalized functions,

$$\int e^{itx}dt = 2\pi\delta(x).$$

Indeed,

$$\begin{aligned}\int dx\, f(x) \int e^{itx}dt &= \sqrt{2\pi}\int dt\, \frac{1}{\sqrt{2\pi}}\int f(x)e^{itx}dx \\ &= \sqrt{2\pi}\int \check{f}(t)\,dt \\ &= 2\pi\cdot\frac{1}{\sqrt{2\pi}}\int \check{f}(t)e^{-it0}dt = 2\pi f(0).\end{aligned}$$

Q: Are these nice functions the same as the nice functions mentioned earlier.

A: Yes, they are.

2.3 Exponential operators

The exponential e^O of an operator O over a topological vector space is the operator

$$e^O = \sum_{k=0}^{\infty}\frac{O^k}{k!} = I + O + \frac{1}{2}O^2 + \frac{1}{6}O^3 + \ldots$$

If $(X\psi)(x) = x\cdot\psi(x)$ then $(e^X\psi(x)) = e^x\psi$, because

$$(e^X\psi)(x) = \sum_{k=0}^{\infty}\frac{1}{k!}(X^k\psi)(x) = \psi\cdot\sum_{k=0}^{\infty}\frac{1}{k!}x^k = \psi\cdot e^x.$$

If D is the derivative operator $\frac{d}{dx}$, then $e^{aD}\psi(x) = \psi(x+a)$. Indeed,

$$\begin{aligned}e^{aD}\psi(x) &= \sum_{k=0}^{\infty}\frac{(aD)^k\psi(x)}{k!} = \sum_{k=0}^{\infty}\frac{D^kf(x)}{k!}a^k \\ &= \psi(x) + \frac{\psi'(x)}{1!}a + \frac{\psi''(x)}{2!}a^2 + \frac{\psi'''(x)}{3!}a^3 + \ldots\end{aligned}$$

which is the Taylor series of $\psi(x+a)$ around point x. (Think of a as Δx.)

Q: What functions f are you talking about? The Taylor series expansion of f suggests that f is analytic, that is real-analytic.

A: Our intention is that f ranges over L^2. By the proof above, e^{aD} is a shift $f(x) \mapsto f(x+a)$ on analytic functions. In particular, e^{aD} is a shift on Gaussian functions

$$\exp\left(-\frac{(x-b)^2}{2c^2}\right).$$

But Gaussian functions span a dense subspace of $L^2(\mathbb{R})$, and there is a unique continuous extension of e^{aD} to L^2, namely the shift $f(x) \mapsto f(x+a)$.

Q: The exponential e^O has got to be a partial operator in general.

A: Yes, $e^O(x)$ is defined whenever the oprators O^k are defined at x, and the series $\sum_{k=0}^{\infty} \frac{O^k(x)}{k!}$ converges.

3 Joint-to-Marginal Lemma

Let $f(x,p)$ be an ordinary probability distribution or a quasi-distribution on $\mathbb{R}^2$. For any $z = ax + bp$ where a, b are not both zero, the marginal distribution $g(z)$ of z can be defined thus:

$$g(z) = \begin{cases} \frac{1}{b}\int f(x, \frac{1}{b}(z-ax))\,dx & \text{if } b \neq 0 \\ \frac{1}{a}\int f(\frac{1}{a}(z-by), p)\,dp & \text{otherwise} \end{cases}$$

Here's a justification in the case $b \neq 0$. We have

$$\begin{aligned} p &= \frac{1}{b}(z-ax), \\ dp &= \frac{1}{b}(dz - a\,dx), \\ f(x,p)\,dx\,dp &= f(x, \frac{1}{b}(z-ax))\frac{1}{b}\,dx\,dz. \end{aligned}$$

We are relying here on the formalism of differential 2-forms [7] for area elements, so that $dx\,dz$ really means $dx \wedge dz$ and we have used that $dx \wedge dx = 0$. The use of differential forms makes computations like this easier, and it fits well with physics, e.g., with Maxwell's equations and with general relativity. One could, however, avoid differential forms here and get the same result by considering the Jacobian determinant of the change of variables.

For any real $u \le v$, the probability that $u \le z \le v$ should be

$$\begin{aligned}\int_u^v g(z)dz &= \iint_{u\le ax+bp\le v} f(x,p)\,dx\,dp \\ &= \iint_{u\le ax+bp\le v} \frac{1}{b} f(x, \frac{1}{b}(z-ax))\,dx\,dz \\ &= \int_u^v dz \int_{-\infty}^{\infty} \frac{1}{b} f(x, \frac{1}{b}(z-ax))\,dx.\end{aligned}$$

Since the first and last expressions coincide for all $u \le v$, we have

$$g(z) = \frac{1}{b}\int f(x, \frac{1}{b}(z-ax))\,dx.$$

Lemma 2 (J2M)**.** *For any a, b not both zero, the following statements are equivalent.*

1. *$g(z)$ is the marginal distribution of $z = ax + bp$.*
2. *$\hat{g}(\zeta) = \sqrt{2\pi}\cdot \hat{f}(a\zeta, b\zeta)$.*

Proof. To prove (1)→(2), suppose (1) and compare the forward Fourier transforms of g and f:

$$\begin{aligned}\hat{g}(\zeta) &= \frac{1}{\sqrt{2\pi}}\int g(z)e^{-i\zeta z}\,dz \\ &= \frac{1}{\sqrt{2\pi}}\iint f(x, \frac{1}{b}(z-ax))e^{-i\zeta z}\frac{1}{b}\,dx\,dz \\ &= \frac{1}{\sqrt{2\pi}}\iint f(x,p)e^{-i\zeta(ax+bp)}\,dx\,dp. \\ \hat{f}(\xi,\eta) &= \frac{1}{2\pi}\iint f(x,p)e^{-i(\xi x+\eta p)}\,dx\,dp.\end{aligned}$$

We have $\hat{g}(\zeta) = \sqrt{2\pi}\hat{f}(a\zeta, b\zeta)$.

To prove (2)→(1), suppose (2) and use the implication (1)→(2). If h is the marginal distribution of $z = ax + by$ then

$$\hat{h}(\zeta) = \sqrt{2\pi}\cdot\hat{f}(a\zeta, b\zeta) = \hat{g}(\zeta),$$

and therefore $g = h$. □

Corollary 3. *For any real α, β not both zero, $\hat{f}(\alpha,\beta) = \frac{1}{\sqrt{2\pi}}\hat{g}(\zeta)$ where $g(z)$ is the marginal distribution for the linear combination $z = ax + bp$ such that $\alpha = a\zeta$, $\beta = b\zeta$ for some ζ.*

4 Wigner uniqueness

The purpose of this section is to prove the Wigner Uniqueness proposition. For simplicity we work with one particle moving in one dimension, but everything we do in this section generalizes in a routine way to more particles in more dimensions.

In classical mechanics, the position x and momentum p of the particle determine its current state. The set of all possible states is the phase space of the particle. By Corollary 3, an ordinary distribution $f(x, p)$ on the phase space is uniquely determined by its marginal distributions for all linear combinations $ax + bp$ where a, b are not both zero.

In the quantum case, a state of the particle is given by a normalized (to norm 1) vector $|\psi\rangle$ in $L^2(\mathbb{R})$. The position and momentum are given by Hermitian operators X and P where

$$(X\psi)(x) = x \cdot \psi(x) \quad \text{and} \quad (P\psi)(x) = -i\hbar \frac{d\psi}{dx}(x).$$

For any a, b not both zero, the linear combination $z = ax + by$ is given by the Hermitian operator $Z = aX + bP$. In a state $|\psi\rangle$, there is a probability distribution $g(z)$ (for the measurement) of the values of z. For a function $h(z)$ of z, the expectation of $h(z)$ is $\langle\psi|h(Z)|\psi\rangle$.

The following technical lemma plays a key role in our proof of the uniqueness of Wigner's quasi-distribution.

Lemma 4.

$$\langle\psi|e^{-i(\alpha X+\beta P)}|\psi\rangle = e^{i\alpha\beta\hbar/2} \int \psi^*(y) e^{-i\alpha y} \psi(y - \beta\hbar)\, dy.$$

Proof. We want to split the exponential into a factor with X times a factor with P. This is not as easy as it might seem, because X and P don't commute. We have, however, two pieces of good luck. First, there is Zassenhaus's formula, which expresses the exponential of a sum of non-commuting quantities as a product of (infinitely) many exponentials, beginning with the two that one would expect from the commutative case, and continuing with exponentials of nested commutators:

$$e^{A+B} = e^A e^B e^{-\frac{1}{2}[A,B]} \cdots,$$

where the "$\cdots$" refers to factors involving double and higher commutators.

> **Q:** You gave no reference to Zassenhaus's paper.
>
> **A:** Apparently, Zassenhaus never published this result, but there's a paper [4] that shows how to compute the next terms. It also has a pointer to early uses of the formula.

The second piece of good luck is that $[X, P] = i\hbar I$, where I is the identity operator. (In the future, we'll usually omit writing I explicitly, so we'll regard this commutator as the scalar $i\hbar$.) Since that commutes with everything, all the higher commutators in Zassenhaus's formula vanish, so we can omit the "$\cdots$" from the formula. We have

$$\langle\psi|e^{-i\alpha X - i\beta P}|\psi\rangle = \langle\psi|e^{-i\alpha X}e^{-i\beta P}e^{\alpha\beta[X,P]/2}|\psi\rangle.$$

The last of the three exponential factors here arose from Zassenhaus's formula as

$$-\frac{1}{2}[-i\alpha X, -i\beta P] = \frac{1}{2}\alpha\beta[X, P] = i\alpha\beta\hbar/2.$$

That factor, being a scalar, can be pulled out of the bra-ket. Taking into account §2.3,

$$\langle\psi|e^{-i(\alpha X+\beta P)}|\psi\rangle = e^{i\alpha\beta\hbar/2}\int \psi^*(y)e^{-i\alpha y}\psi(y-\beta\hbar)\,dy.$$

□

Now we are ready to prove the Wigner Uniqueness proposition. Suppose that a quasi-distribution $f(x, p)$ yields correct marginal distributions for all linear combinations of position and momentum. For any real α, β not both zero, let a, b, g, ζ be as in Corollary 3. Then

$$\begin{aligned}\hat{f}(\alpha,\beta) = \frac{1}{\sqrt{2\pi}}\hat{g}(\zeta) &= \frac{1}{2\pi}\int g(z)e^{-i\zeta z}\,dz \\ &= \frac{1}{2\pi}\langle e^{-i\zeta Z}\rangle = \frac{1}{2\pi}\langle\psi|e^{-i\zeta Z}|\psi\rangle \\ &= \frac{1}{2\pi}\langle\psi|e^{-i\zeta(aX+bP)}|\psi\rangle.\end{aligned} \tag{1}$$

By Lemma 4,

$$\hat{f}(\alpha,\beta) = \frac{e^{i\alpha\beta\hbar/2}}{2\pi}\int \psi^*(y)e^{-i\alpha y}\psi(y-\beta\hbar)\,dy. \tag{2}$$

To get $f(x, p)$, apply the (two-dimensional) inverse Fourier transform.

$$f(x,p) = \frac{1}{(2\pi)^2}\iiint \psi^*(y)e^{-i\alpha y}e^{i\alpha\beta\hbar/2}\psi(y-\beta\hbar)e^{i\alpha x}e^{i\beta p}\,dy\,d\alpha\,d\beta.$$

Collecting the three exponentials that have α in the exponent, and noting that α appears nowhere else in the integrand, perform the integration over α and (recall §2.2) get a Dirac delta function:

$$\int e^{-i\alpha(y-\frac{\beta\hbar}{2}-x)}\,d\alpha = 2\pi\delta(y - x - \frac{\beta\hbar}{2}).$$

That makes the integration over y trivial, and what remains is

$$f(x,p) = \frac{1}{2\pi}\int \psi^*(x+\frac{\beta\hbar}{2})\psi(x-\frac{\beta\hbar}{2})e^{i\beta p}\,d\beta, \tag{3}$$

which is Wigner's quasi-distribution.

To check that Wigner's quasi-distribution yields correct marginal distribution note that the derivation of (3) from (1) is reversible. This completes the proof of the Wigner Uniqueness proposition.

5 Weyl's correspondence

There is another approach to characterizing the Wigner quasi-probability distribution, using the expectation values for a wide class of functions rather than the marginal distributions for just the linear functions of position and momentum. The main ingredient for this approach is a proposal by Hermann Weyl [19, §IV.14] for associating a Hermitian operator on $L^2(\mathbb{R})$ to any (well-behaved) function $g(x,p)$ of position and momentum. Weyl's proposal is to first form the Fourier transform $\hat{g}(\alpha,\beta)$ of $g(x,p)$, and then apply the inverse Fourier transform with the Hermitian operators X and P in place of the classical variables x and p. Thus, the Weyl correspondence associates to $g(x,p)$ the operator

$$g(X,P) = \frac{1}{2\pi}\iint \hat{g}(\alpha,\beta)e^{i(\alpha X+\beta P)}\,d\alpha\,d\beta.$$

If one grants that this is a reasonable way of converting phase-space functions $g(x,p)$ to operators $g(X,P)$, then a desirable property of a phase-space quasi-probability distribution $f(x,p)$ would be that the expectation of $g(X,P)$ in a quantum state $|\psi\rangle$ is the same as the expectation of $g(x,p)$ under $f(x,p)$. We shall show that the Wigner distribution is uniquely characterized by enjoying this desirable property for all well-behaved g.

Indeed, the expectation of $g(X,P)$ in state $|\psi\rangle$ is

$$\langle\psi|g(X,P)|\psi\rangle = \frac{1}{2\pi}\iint \hat{g}(\alpha,\beta)\langle\psi|e^{i(\alpha X+\beta P)}|\psi\rangle\,d\alpha\,d\beta,$$

and the expectation of $g(x,p)$ under the distribution $f(x,p)$ is

$$\iint g(x,p)f(x,p)\,dx\,dp = \iint \hat{g}(\alpha,\beta)\hat{f}(\alpha,\beta)\,d\alpha\,d\beta.$$

This last equation is a consequence of the fact, mentioned in §2.1, that the Fourier transform is a unitary operator and therefore preserves the inner product structure of $L^2(\mathbb{R})$. Since these two expectations agree for all (well-behaved) g,

$$\hat{f}(\alpha,\beta) = \frac{1}{2\pi}\langle\psi|e^{i(\alpha X+\beta P)}|\psi\rangle.$$

But this is the part of equation(1) that was used to derive Wigner's formula (3).

> **Q:** I wonder how Weyl arrived at his proposal.
>
> **A:** Weyl presents his proposal in [19] without any motivation, so we don't know how he came up with it, but we can speculate. The title of [19] indicates that Weyl was working in a group-theoretic context. As a result, the Fourier transform, expressing functions on $\mathbb{R}$ as combinations of the characters $e^{i\alpha x}$ of the group $(\mathbb{R}, +)$ would be in the forefront of his considerations. Now consider his goal — to somehow convert a classical function $g(x, p)$ into an operator. Roughly speaking, he would want to substitute the operators X and P for the classical variables x and p. An obvious difficulty is that the same functions $g(x, p)$ might have two different expressions, for example $xp = px$, which are no longer equivalent when operators are substituted, $XP \neq PX$. So it is reasonable to try to choose, from the many expressions for a function $g(x, p)$, one particular, reasonably canonical expression, into which one can substitute X and P. The Fourier expansion, $\int \hat{g}(\alpha,\beta) \exp(i(\alpha x+\beta p))\, dx\, dp$ has those properties. It depends only on the function g, not on how one chooses to express it, and there is no problem substituting X and P for x and p.

6 Feynman and spins

Richard Feynman studied "an analogue of the Wigner function for a spin $\frac{1}{2}$ system or other two state system" [6]. He chose the z and x components of the spin to serve as the analogs of the position and momentum in Wigner's formula.

Q: His case should be much simpler than Wigner's case.

A: Not necessarily. While the commutator $[X, P]$ is a scalar, the commutator of the z and x components of a spin is the y-component times $i\hbar$. This is but one of several complications.

Q: Is Feynman's quasi-distribution determined by the correct marginals for all linear combinations of the x and z spins?

A: That question sounds reasonable until you look at it a little more closely. To fix notation, let's describe spin by means of the standard Pauli matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The usual matrix representation of the spins for a spin $\frac{1}{2}$ particle is given by these matrices divided by 2, but it's convenient to skip those extra factors $\frac{1}{2}$; if you like, imagine that we are measuring angular momentum in units of $\hbar/2$ instead of $\hbar$.

So each of our matrices has eigenvalues ± 1, with $+1$ meaning spin along the corresponding positive axis and -1 along the corresponding negative axis. For example, the two basis states of spin up and spin down along the z axis are the eigenvectors for eigenvalues 1 and -1 of Z; equivalently, they correspond to eigenvalues 1 and 0 for $(I + Z)/2$, where I is the identity operator. This point of view is useful because it implies that, in any state $|\psi\rangle$, $(1 + \langle Z\rangle)/2$ is the probability that the z spin is up. Here, as before, angle brackets denote expectations. Similarly, $(1 - \langle Z\rangle)/2$ is the probability that the z spin is down. Of course, analogous formulas apply to the x and y components of the spin.

Feynman, in analogy to Wigner, introduces a quasi-probability distribution f for the pair of non-commuting observables Z and X. So f has four components, f_{++}, f_{+-}, f_{-+}, and f_{--}, as the quasi-probability of Z and X having the values ± 1 given by the subscripts of f.

Now let's look at a linear combination of Z and X, for example the simplest nontrivial one, $Z + X$. From the point of view of quasi-probabilities,

- with probability f_{++}, Z has value 1 and X has value 1, so $Z + X$ has value 2,
- with probability f_{+-}, Z has value 1 and X has value -1, so $Z + X$ has value 0,
- with probability f_{-+}, Z has value -1 and X has value 1, so $Z + X$ has value 0, and
- with probability f_{--}, Z has value -1 and X has value -1, so $Z + X$ has value -2.

Altogether, the possible values of $Z + X$ are 2, 0, and -2, with quasi-probabilities f_{++}, $f_{+-} + f_{-+}$, and f_{--}, respectively.

So your proposed analog of the result for Wigner's distribution would assume that f is chosen so that these probabilities agree, in a given state, with the probabilities computed by quantum mechanics. But such agreement is impossible, because, according to quantum mechanics, the possible values of $Z + X$ are the eigenvalues of this operator, namely $\pm\sqrt{2}$, which are completely different from the $2, 0, -2$ arising from the quasi-probabilities. It is easy to check that the possible values of any nontrivial linear combination of Z and X are completely different from the values arising from the quasi-probabilities.

Q: OK, let's require the minimum that Feynman obviously intended, namely that f should produce the correct marginal distributions of Z and X. Does this determine f uniquely?

A: At first sight, this looks promising. Requiring the correct marginals for the two variables, each having two possible values, gives us four equations for the four

unknown components $f_{\pm\pm}$ of f:

$$f_{++} + f_{+-} = \frac{1}{2}(1 + \langle Z \rangle)$$
$$f_{-+} + f_{--} = \frac{1}{2}(1 - \langle Z \rangle)$$
$$f_{++} + f_{-+} = \frac{1}{2}(1 + \langle X \rangle)$$
$$f_{+-} + f_{--} = \frac{1}{2}(1 - \langle X \rangle).$$

But there's redundancy in the equations; only three of them are independent, so there's one free parameter in the general solution. In fact, it's easy to write down the general solution:

$$f_{++} = \frac{1}{4}(1 + \langle Z \rangle + \langle X \rangle + t)$$
$$f_{+-} = \frac{1}{4}(1 + \langle Z \rangle - \langle X \rangle - t)$$
$$f_{-+} = \frac{1}{4}(1 - \langle Z \rangle + \langle X \rangle - t)$$
$$f_{--} = \frac{1}{4}(1 - \langle Z \rangle - \langle X \rangle + t),$$

where t is arbitrary. Feynman's formulas correspond to $t = \langle Y \rangle$ but we see no reason to prefer $\langle Y \rangle$ over, for example, $-\langle Y \rangle$.

Q: Put the freedom in choosing t to some use. How about minimizing the negativity in f? In other words, adjust t to bring f as close as possible to being a genuine probability distribution.

A: That idea works better than we originally expected. One can get rid of the negativity altogether. For each state $|\psi\rangle$, there is a choice of t that makes all four components of f nonnegative.

Indeed, write down the four inequalities $f_{\pm\pm} \geq 0$ using the formulas above for these $f_{\pm\pm}$'s. Solve each one for t. You find two lower bounds on t, namely

$$-1 - \langle Z \rangle - \langle X \rangle \text{ (from } f_{++} \geq 0)$$
$$-1 + \langle Z \rangle + \langle X \rangle \text{ (from } f_{--} \geq 0),$$

and two upper bounds, namely

$$1 + \langle Z \rangle - \langle X \rangle \text{ (from } f_{+-} \geq 0)$$
$$1 - \langle Z \rangle + \langle X \rangle \text{ (from } f_{-+} \geq 0).$$

An appropriate t exists if and only if both of the lower bounds are less than or equal to both of the upper bounds. That gives four inequalities, which simplify to $-1 \leq \langle Z \rangle \leq 1$ and $-1 \leq \langle X \rangle \leq 1$. But these are always satisfied, because the eigenvalues of Z and X are ± 1.

Q: I am confused. The uncertainty principle asserts that you cannot measure Z and X at once. Accordingly one would expect that the joint probability distribution $f_{\pm\pm}$ should not exist or, as in Wigner's case, should have at least one negative value.

A: The relevant difference between Wigner's and Feynman's cases seems to be this. In Wigner's case, there is naturally a rich set of marginals that the joint probability distribution is supposed to produce, namely the probability distributions of all linear combinations of the position x and the momentum p of the particle. In Feynman's case, the natural set of marginals is too poor, just the probability distributions of Z and X.

Q: Did Feynman find a good use for quasi-probabilities?

A: He introduced negative probabilities in connection to a problem of infinities in quantum field theory. "Unfortunately I never did find out how to use the freedom of allowing probabilities to be negative to solve the original problem of infinities in quantum field theory!" [6].

Q: Still, the idea to use quasi-probability distributions to simplify intermediate computations looks attractive to me.

A: You are in good company.

References

[1] Samson Abramsky, "Contextual semantics: From quantum mechanics to logic, databases, constraints, and complexity," in Logic in Computer Science Column, Bulletin of EATCS 113, 26 pages (2014).

[2] George A. Baker, Jr., "Formulation of quantum mechanics based on the quasi-probability distribution induced on phase space," Physical Review 109:6 (1958) 2198–2206. Reprinted in [22], 235–243.

[3] Jacqueline Bertrand and Pierre Bertrand, "A tomographic approach to Wigner's function," Foundations of Physics 17:4 (1987) 397–405.

[4] Fernando Casas, Ander Murua, and Mladen Nadinic, "Efficient computation of the Zassenhaus formula," Computer Physics Communications 183 (2012) 2386–2391.

[5] William Feller, *An Introduction to Probability Theory and Its Applications, Volume 1*, John Wiley (1950), 3rd edition (1968).

[6] Richard P. Feynman, "Negative probabilities," in *Quantum Implications: Essays in Honor of David Bohm*, ed. F. D. Peat and B. Hiley, Routledge & Kegan Paul (1987) 235–248. Reprinted in [22], 426–439.

[7] Harley Flanders, *Differential Forms with Applications to the Physical Sciences,* Academic Press (1963). 2nd edition, Dover (1989).

[8] Brian C. Hall, *Quantum Theory for Mathematicians,* Springer-Verlag, Graduate Texts in Mathematics 267 (2013).

[9] John F. C. Kingman and S. James Taylor, *Introduction to Measure and Probability,* Cambridge Univ. Press (1966).

[10] José E. Moyal, "Quantum mechanics as a statistical theory," Proc. Cambridge Phil. Soc. 45 (1949) 99–124. Reprinted in [22], 167–192.

[11] G. Nogues, A. Rauschenbeutel, S. Osnaghi, P. Bertet, M. Brune, J. M. Raimond, S. Haroche, L. G. Lutterbach, and L. Davidovich, "Measurement of a negative value for the Wigner function of radiation," Physical Review A 62, 054101 (2000).

[12] Robert F. O'Connell and Eugene P. Wigner, "Quantum-mechanical distribution functions: conditions for uniqueness," Physics Letters 83A:4 (1981), 145–148.

[13] Alexei Ourjoumtsev, Rosa Tualle-Broui, and Philippe Grangier, "Quantum homodyne tomography of a two-photon Fock state," arXive:quant-ph/0603284 (2006).

[14] Walter Rudin, *Real and Complex Analysis,* McGraw-Hill (1966). 3rd edition (1987).

[15] D. T. Smithey, M. Beck, M. G. Raymer, and A. Faridani, "Measurement of Wigner distribution and the density matrix of a light mode using optical homodyne tomography," Physical Review Letters 70:9 1244-1247 (1993).

[16] Robert W. Spekkens, "Negativity and contextuality are equivalent notions of nonclassicality," arXiv:0710.5549v2 (2008).

[17] Victor Veitch, Christopher Ferrie, David Gross, and Joseph Emerson, "Negative quasi-probability as a resource for quantum computation," arXiv:1201.1256 (2012).

[18] Victor Veitch, S. A. Hamed Mousavian, Daniel Gottesman, and Joseph Emerson, "The resource theory of stabilizer quantum computation," New Journal of Physics 16 013009 (Jan. 9, 2014).

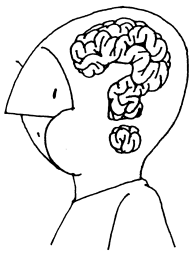
[19] Hermann Weyl, *Quantenmechanik und Gruppentheorie*, Zeitschrift für Physik 46 1–46. Reprinted in [22], 45–90.

[20] Eugene P. Wigner, "On the quantum correction for thermodynamic equilibrium," Physical Review 40 (1932) 749–759. Reprinted in [22], 100–110.

[21] Eugene P. Wigner, "Quantum mechanical distribution functions revisited," in W. Yourgrau and A. van der Merwe (eds), *Perspective in Quantum Theory,* MIT Press, MA, 1971) 25–36 (1971).

[22] Cosmas K. Zachos, David B. Fairlie, and Thomas L. Curtright (eds.), *Quantum Mechanics in Phase Space: An Overview with Selected Papers,* World Scientific Series in 20th Century Physics, vol. 34 (2005).

News and Conference Reports



News from New Zealand

BY

C. S. Calude

Department of Computer Science, University of Auckland
Auckland, New Zealand
cristian@cs.auckland.ac.nz

After 21 years of uninterrupted presence (this column didn't appear in the October 2013 issue due to a clerical problem in the change of editors-in-chief) this is the last report from New Zealand.

I wish to warmly thank Professor G. Rozenberg for inviting me to contribute with a column of news to the Bulletin, the other Bulletin editors-in-chief I have worked with, the eminent scientists who have accepted to be interviewed, and, last but not least, the Bulletin readers.

This column includes the continuation of the interview with Professor G. Rozenberg. An edited-augmented-structured selection of interviews published in this column will appear in the book *The Human Face of Computing* to this year at Imperial College Press, London.

1 Scientific and Community News

The latest CDMTCS research reports are (`http://www.cs.auckland.ac.nz/staff-cgi-bin/mjd/secondcgi.pl`):

470. B. Khoussainov . A Quest For Algorithmically Random Infinite Structures, II

2 A Dialogue with Grzegorz Rozenberg about Natural Computing II

Professor G. Rozenberg is a professor at the Leiden Institute of Advanced Computer Science of Leiden University, The Netherlands and adjoint professor at the Department of Computer Science, University of Colorado at Boulder, USA. He published over 500 papers, 6 books, and is a (co-)editor of more than 100 books on natural computing, formal language and automata theory, graph transformations, and concurrent systems. He founded a number of journals and book series in theoretical computer science and natural computing. He is often referred to as the guru of natural computing.

Professor Rozenberg is a Foreign Member of the Finnish Academy of Sciences and Letters, a member of Academia Europaea, and he is holder of Honorary Doctorates of the University of Turku, Finland, the Technical University of Berlin, Germany, the University of Bologna, Italy, and Åbo Akademi, Swedish University in Turku, Finland. He has received the Distinguished Achievements Award of the EATCS.

With the artist name Bolgani, he is a performing magician specialising in close-up illusions.

CC: What is more important for research in natural computing: potential applications or understanding how “nature" computes? Does “nature" really compute?

GR: Advancing our understanding of the computation taking place in nature will often lead to (potential) advances in human-designed computing, simply because so often the way nature computes is superior to the way humans compute. In this sense, understanding how nature computes is “primary”. But there is a catch here. Our classical notion of computation is rooted in the quest for formalising

the way humans compute/calculate—it dates back (at least) to the work of Leibniz and it culminated in the first half of the 20th century with the research/results by Post, Church, and Turing. Quite often, this (beautiful) notion/idea of computation does not really apply to the computation going all around us in nature because it violates various "underlying axioms" of the way that nature works. I strongly believe that research in natural computing will eventually lead to a novel notion of computation, as a matter of fact to a new "science of computation" which will be developed by the interaction/co-operation of computer scientists, biologists, chemists, mathematicians, physicists, ... Indeed, the research in natural computing has already changed our understanding of what computing is about.

CC: Can you give examples of such differences in underlying axioms?

GR: An answer to such a question should be given by writing a series of papers but let me just mention one such difference, *persistence*.

In models of computation in computer science one assumes that if in a global state a local part of it is not "touched", then this local part will be preserved, i.e., it will be a local component of the successor global state. This does not hold in biology, e.g., when you model the living cell. An entity from a current state will be present in the successor state only if it is produced, hence sustained, by a reaction (or "thrown in" by the environment). This reflects a basic principle of bioenergetics: life must be sustained. Standard computer science models of computation would imply immortality!

CC: You founded two journals and a book series dedicated to natural computing. Tell us more about them.

GR: The journals are: "Natural Computing", originally published by Kluwer and then taken over by Springer, and "Theoretical Computer Science, Series C: Theory of Natural Computing " (TCSC) by Elsevier, and the book series is "Natural Computing " by Springer. "Natural Computing " is a journal of a very broad scope: it covers experimental, applied and theoretical aspects of natural computing. There you will find publications by biologists, chemists, nano-scientists, physicists, mathematicians, computer scientists, ...—it really reflects the genuinely interdisciplinary nature of natural computing. It is an ideal journal for publishing special issues of interdisciplinary conferences such as, e.g., "DNA Computing". The TCSC journal, on the other hand, aims at publishing theoretical papers that are in the style of the well established Theoretical Computer Science journal. The book series "Natural Computing " by Springer publishes both texts and monographs covering the whole spectrum of "natural computing" (theory, experiments, and applications). All three publications were very well received by the scientific community. They are doing very well and will certainly grow and flourish in the years to come.

CC: You edited four influential handbooks. The last one is "Handbook of Nat-

ural Computing " which is really huge; tell us more about it.

GR: It is indeed huge: 4 volumes, around 2100 pages, over 100 contributing authors, ... This was the editing project which, by far, took most of my time and energy. But I am happy with the result.

The goal of the Handbook is two-fold: (1) to provide an authoritative reference for a significant and representative part of the research in natural computing, and (2) to provide a convenient gateway to natural computing for motivated newcomers to this field.

Apparently, we have succeeded, as the handbook seems to be popular among both groups of readers; also, it has received very good reviews.

Sometimes one compares writing a book to writing a symphony and editing a book to directing an orchestra. I think that this is a nice and fitting comparison, except that there are two important differences between editing a book and directing an orchestra. Usually there is a rehearsal period (often intense) before an actual performance by an orchestra and mostly the individual players do follow very closely (to the best of their abilities) the instructions of the conductor. Unfortunately in book editing (especially with a large number of authors, like, e.g., in a handbook) there is essentially no rehearsal and close following of the instructions by the editor is very difficult to enforce.

CC: You are also a magician ...

GR: I am a performing magician, but I earn my living as a scientist. Both science and magic are beautiful and provide an exciting way of living. I feel very privileged that an interleaving of these two strands of creativity forms the double helix which determines my creative life.

Although science is very rational and magic is emotional, there are many similarities between them. Here are some similarities:
(1) First of all, both are based on creativity – the main source of success in both disciplines.
(2) An important lesson you learn from magic (either as a performer or as a spectator) is NOT to accept things on their face value (you just saw the King of Hearts in a deck of cards, but when you inspect the deck this card is not there!). Thus you need to question everything, which in fact is one of the key principles of an original research in science.
(3) Trying to achieve something astonishing/impossible is one of the key motives/incentives in both science and magic. In mathematics (theoretical computer science) we get a great satisfaction if we settle a conjecture (preferably an "old" conjecture), because in this way we achieve something that was difficult/impossible for other scientists. Perhaps, the satisfaction is even greater when we disprove a conjecture, as then we are even closer to something impossible (something believed to be not true). Similarly an important goal/essence of out-

standing magic is to get as close as possible (close by epsilon) to something totally impossible, something that contradicts the reality (as we know it). My (magician's) business card says "Be Astonished by The Impossible."

However, there is also a cultural difference between scientists and magicians concerning achieving something "impossible". In my long scientific career I witnessed too often situations where scientists (too quickly) declared: "This looks impossible, let's do something else." On the other hand, in my long life in magic I heard quite often a statement of this sort: "This looks impossible, let's work on it".

There are also other important differences – here is one of them. Magic is a performing art, and the performance is the essence of magic. In science the standing/quality of a scientist is determined by her/his peers (e.g., the quality/acceptance of your publications is determined by reviews done of your peers). In magic there are essentially two ways in which your quality is determined/judged.
(i) The first one, the primary one, is the judgement by spectators – you are a good/great magician if this is the judgement by your audience (of laymen).
(ii) Then there is a judgement by your peers, e.g., when you demonstrate/discuss magic in a magic club. The judgement of your peers may be very different, e.g., they may be fascinated by your mastery of a specific sleight-of-hand, while you may be a lousy performer. But . . . it is a performance that determines an emotional reaction of spectators – without it a lot of magic would be reduced to various kinds of puzzles!!!

As a matter of fact, the attitude of spectators is one of the problems facing magicians. Quite often, spectators (especially scientists or those who see a real magic performance for the first time) come to a show with the "puzzle attitude" – they sit there totally stressed, watching your every move, ready to catch you, to solve the puzzle. I always explain at the beginning of a show that this is a wrong attitude because illusions have no explanation and so there is nothing there to discover (to catch onto). In fact, if they would ever get a glimpse of "something", then this particular effect was not an illusion. When I explained this to Mike Rabin (after performing for him and his wife) he called this "The Rozenberg Principle": an event observed disappears! Thus, there is a direct link between magic and quantum mechanics! Apropos, I have a magic show focussed on explaining some of the principles behind various areas of natural computing – one of the card effects demonstrates some principles of quantum computing.

For several years now, many news items in a variety of media (newspapers, scientific journals, social media) suggest that magic is just misdirection. I get often links to such news items from my friend Moshe Vardi (and he expects my comments). For example, some prestigious neuroscience journals publish studies which demonstrate how misdirection by a good magician (and some very good

magicians are involved in these studies) can fool our brains and what it implies for the understanding of the functioning of the brain. Although some of these studies are interesting, a big flaw (in my opinion) is that the involved neuroscientists become convinced that they have become magicians (through conducting these experiments), and hence express opinions as if they were magicians, while the involved magicians become convinced that they have become neuroscientists (through their participation in these experiments) and hence express opinions as if they were neuroscientists. As a result, the studies/conclusions presented are often superficial. I wrote several times to Moshe that some beautiful magic effects indeed rely on misdirection, but on the other hand some beautiful magic effects have nothing to do with misdirection. You may be a top magician without ever employing misdirection – "magic by misdirection" is just a part of magic, in the same way as graph theory or calculus are parts of mathematics.

Living in two worlds, science and magic, is also enriching from the social point of view. Through my life in magic I got embedded in a wonderful community of people (which is very different from the academic community). The notion of quality (of magic performance) is central here and it is also easy to interact with other magicians. Once you have something interesting/amazing to show, a whole stream of interactions follows: you get comments, your spectators (magicians) show you their creations, ... naturally very close contacts are established and they often lead to collaborations and friendships.

CC: How did your family react to your "magic"?

GR: When my son Daniel was a teenager, he was not really impressed by the fact that I was a university professor. But the fact that I was a magician was a different story. Once, when I got back home from my office and entered our house I found Daniel and his friend Ferdie in the hall. After Daniel introduced me to Ferdie I proceeded to the kitchen to have a glass of water. While drinking water I heard Ferdie asking Daniel "What is your father's profession?" to which Daniel replied "He is a university professor" and then a few seconds later Daniel added "But he is not stupid, he is a very good magician"! Also, my grandson Mundo is now very proud that his grandpa is a wizard.

Coincidence, predestination, ... are important concepts in magic. Thus I wonder (magic, unlike mathematics, does not have to be rational) whether being a magician was my predestination. First of all, the maiden name of my mother was Zauberman (which translates into "magician") – I did not notice this connection until I was already in my thirties. Then, my wife's name is Maja and during a trip to India we were told that in Hindu Maja means "illusion". It is certainly great for a magician to be married to an illusion – this makes my magician friends pretty jealous!!!

CC: How do you see computer science after working in the field for about 50

years?

GR: The visibility and importance of computer science grew very impressively during this period. The main reason is the spectacular progress in Information and Communication Technology (ICT) which is very much driven by progress in computer science. This is a blessing but also a curse for computer science, because computer science is often perceived, by the public as well as by scientists from other areas of science, as a technological discipline, a collection of practical skills. Perhaps the most frequent perception of a computer scientist is that of a skilful programmer, an educated hacker. I remember that a long time ago at my university in Leiden, some physicists were not supporting a formation of a computer science department because "our people are also good programmers".

However there is so much more to computer science than ICT. The only reasonable definition of computer science is that this is THE science of information processing. If you consider a typical computer science department and observe the specialities of its faculty members, you will get a list of this sort: computer graphics, data bases, human-computer interactions, natural language processing, computer architecture, programming languages, theory of computation, compiler construction, bioinformatics, concurrent systems, ... The only common denominator for all these research areas is that they are concerned with various aspects of information processing. As a matter of fact the term "Informatics" used in Europe is much better than "Computer Science" which suggests that computer science is just focussed on one specific device/instrument, viz., computer.

Thus informatics is the science of information processing and it is concerned with information processing in computers and elsewhere, e.g., in nature. Therefore informatics is a fundamental science for other scientific disciplines. This historical evolution of computer science into becoming also a fundamental science of information processing was strengthened by developments of some other scientific disciplines, especially in the second half of the 20th century, which adopted "Information" and "Information Processing" as their central notions and thinking habits. Biology and physics are prime examples of such development – in both areas informatics provides not only instruments but also a way of thinking.

This is not just the opinion of a computer scientist, but also the conviction of top biologists and physicists. For example:

- Richard Dawkins, a famous evolutionary biologist, says "If you want to understand life, don't think about vibrant throbbing gels and oozes, think about information technology".

- Sydney Brenner, one of the best known living biologists, Nobel Prize winner, says "Biology is essentially (very low energy) physics with computation".

- John Wheeler, eminent physicist, stated that while some time ago he thought that everything is particles, now he thinks that everything is information.

Since informatics is THE science of information processing it has a strong interdisciplinary character. As a matter of fact, I remember that in 1971 or 1972 when I was in the department of computer science at the State University of New York at Buffalo a delegation from NSF was visiting there. Tony Ralston, our department chair, asked me (probably as a representative of the "European school") to present my vision of computer science to this delegation. I said then that I envision computer science departments of the future to be divided into groups dealing with the "core computer science", language processing (linguistics), artificial intelligence, biology, physics, ... In particular I was arguing for biology where information processing is so apparent and so challenging to understand. Tony told me later that he heard from the delegation that they did not share my vision and in particular they thought that the relationship between biology and computer science is not as strong/intrinsic as I suggested (in their opinion it was rather superficial). I am glad to conclude that now, over 40 years later, everyone involved must conclude that they were awfully wrong. As a matter of fact it is apparent today that the interdisciplinarity of informatics is one of the main forces driving the tremendous progress of our disciplines.

It is fashionable nowadays to discuss grand challenges of informatics. I am myself convinced that one of the grandest grand challenges of informatics is to understand the world around us in terms of information processing. Each time progress is made in achieving this goal, both the world around us and informatics benefits. Natural computing is a natural avenue of research for achieving such a progress!

CC: Please reminiscence about your youth in Poland.

GR: I grew up in communist Poland, so my youth was dramatically different from the youth of my son in The Netherlands. It was not the best place to grow up, but on the other hand (seen from the perspective of time) in this way I got a deeper understanding of some important issues in life, deeper than that of my friends in my "new world". I strongly believe that deep matters in life can be understood only by experiencing them. As a matter of fact, I am often irritated by the attitude of many intellectuals who make statements of the sort "I understand what it means to live through a terrible war (or to live in a totalitarian system), because I read many books about it".

I received my education in Warsaw, Poland. As everywhere else in the world, the quality of teachers in my schools determined my "initial" taste/liking for many subjects. Thus, I had an awful teacher of chemistry and so I did not like chemistry at all, while today I think that chemistry is relevant, fascinating, and simply beautiful. On the other hand, I had a brilliant teacher of mathematics, his name was

Taytelbaum. He became my idol, and so I had already fallen in love with mathematics at school already. Coming back to the issue of coincidences in magic, Eddy Taytelbaum, one of the nestors of magic in The Netherlands, became my idol and friend quite soon after I got embedded into Dutch magic (to understand this coincidence one has to realize that Taytelbaum is a very uncommon name in Poland and it is a very uncommon name in The Netherlands).

I chose to study electronics at Warsaw University of Technology, as then electronics was then a very modern direction of study, known for its high level of quality (entry exams were very competitive), and, most importantly, in this way I could combine my love for mathematics and physics with my curiosity about technology. It turned out to be a very good choice for me – I found many classes interesting and challenging, and, very importantly for my later life as a researcher (I am theoretician), I got an understanding/feeling of and a respect for engineering.

For my master thesis (for the master degree in computer science) I chose "Theory of algorithms". At that time in Poland this was a combination of Markov algorithms and Turing machines. My interest in the theory of algorithms was instigated through my study of logic circuits design, where a transitive closure of references led me to basic papers on automata theory.

As a matter of fact I got so fascinated by automata theory that, while I was still a student, I approached one of the assistant professors, Pawel Kerntopf (who became a very good friend of mine), and with his help organised a seminar on automata theory. This seminar involved both graduate students and faculty. It turned out to be very successful in many respects. When I recently gave a series of lectures in Warsaw, I was told by colleagues from my Alma Mater that at least 15 participants of my seminar became later professors in Poland and abroad!

While working on my master thesis I met Andrzej Ehrenfeucht from the Mathematical Institute of the Polish Academy of Sciences. Meeting Andrzej changed my life in many ways. He became my source of wisdom on the theory of algorithms (my formal advisor from the department of electronics knew very little about this area). Moreover, it "clicked" between us and very quickly we became friends and later brothers by choice (we consider ourselves brothers). It is because of Andrzej that my love for the technology of information processing changed into love for the theory of information processing. We just celebrated 50 years of scientific cooperation – during this period we wrote hundreds of joint papers and spent thousands of hours talking to each other about scientific and many other matters.

Andrzej is a true renaissance man, deeply knowledgeable about so many areas: mathematics, linguistics, geology, physics, biology, spiders, dinosaurs, fossils, history and teaching of mathematics, ... Often, when I ask Andrzej a question I not only get an answer, I get a whole tutorial. Since 1971 I travel (on average twice a year) to Boulder, Colorado, where I am an adjoint professor in the department

of computer science of the University of Colorado. The main reason for me to travel there is to be with Andrzej. One of the many nice things related to our friendship/brotherhood is that Andrzej loves my magic – nobody else has seen as many of my magic shows, moreover he is the best spectator I ever had.

Andrzej is very much interested in the history of mathematics and in the didactics of mathematics. He collaborates in research in these areas with Pat Bagget, his life partner – she is a professor of mathematics at New Mexico State University in Las Cruces. They are a very nice couple and it is always a pleasure when we three get together.

Telling political jokes in Poland was important for intellectual survival. This was the only way that one could beat the system into pieces. I have created many political jokes, which was pretty dangerous. As a matter of fact when I would tell a friend (whom I could trust) that I have a new joke, then he/she would first ask me "How good is this new joke?" and my typical answer would be "about 5 years" (referring to a punishment, the number of years in prison, in case that I would be "caught" when telling this joke). On my recent lecturing trip to Warsaw, when I met with a group of my colleagues and friends from my years in Poland, I was reminded about my creativity in inventing new jokes, and also reminded how dangerous it was. It was quite interesting for me to learn that some of my jokes are still in circulation today!

CC: Please tell us an "about 5 years" joke.

GR: Here is one. A huge factory was built in a communist country. It was going to serve as a symbol of the superiority of the communist system, thus many visitors came to see it. Important visitors were given a tour of the factory by the mayor of the city. On one of such tours when the visitors arrived at the entrance gate the mayor proudly announced: "This is the biggest factory in the world employing 50,000 workers. It could be built only in a communist country." However, because of the noise of a truck passing by, one visitor didn't hear the first part of the sentence, so he asked: "How many people work in this factory?" The mayor answers: "Oh, you mean *working* here. Perhaps two or three."

This joke describes in a compact way one of the big disasters of the communist system: the destruction of work ethics. For many years after I left Poland I was planning to write a book "The Essence of Communism" which would consist of a set of jokes illuminating various features of the system. Unfortunately, because of chronic shortage of time, this project never materialised.

CC: You started your academic career at the Institute of Mathematics of Polish Academy of Sciences which was one of the world-famous mathematical research institutes ...

GR: Even before I completed my master thesis, I was offered a position at the Institute of Mathematics of Polish Academy of Science (Polish acronym: IM-

PAN) in the group of mathematical logic headed by Andrzej Mostowski. I was also offered a position in the Electronics Department, but I made a choice for IM-PAN because I wanted to pursue research in theory – this was among the best choices I ever made! The senior members of the mathematical logic group were Mostowski, Grzegorczyk, Pawlak and Ehrenfeucht. Mostowski was a very "special" man: very kind with very good manners (a real gentleman) and genuinely friendly. He was very positive during my interview for the position in his group. He had only one small "objection", viz., that I was very young. I still remember when, looking through administrative documents, he said "I see that you will be the youngest member of our group" and then he added "but this problem will resolve itself with time". I recalled this statement many times later in my life as I saw myself to be first the youngest professor, then a well-established member of a department, then a senior professor, and then Professor Emeritus! The kindness of Mostowski manifested itself also in the fact that he always had time when I asked for a consultation in matters of logic. I had less contacts with Grzegorczyk, but I had numerous discussions with him concerning computability theory, especially about his hierarchy of recursive functions. Grzegorczyk wrote a very good book on mathematical logic (in Polish) and I benefited a lot from discussing with him in depth various topics from this book.

I spent a lot of time with Zdzisław Pawlak – we also became very good family friends. He was a wonderful person and a great scientist. He had a very good understanding of the applied aspects of computer science and an extraordinary talent for forming elegant, simple models capturing the essence of applications.

For a man of unusual talents he was very modest. He had a great sense of humour and loved good jokes – his laugh was very contagious. He was one of the few people whom I trusted with my new political jokes. Kayaking and walking were his two favourite physical activities. He had a great talent for writing rhymes and in the later phase of his life he was painting – he was a good painter. His scientific talents are best illustrated by the framework of rough sets which he invented in his sixties. It is an area of research which is very impressive by both its theory and applications, and it is immensely popular all over the world.

He was a delightful friend and I remember that I got very emotional when he told me that I was his best friend.

IMPAN was an "exclusive" institute as so many famous mathematicians worked there. Kuratowski was the director when I worked there. Among other famous mathematicians there were Sierpinski, Łoś, and Sikorski. I had quite frequent contacts with Łoś, but especially with Robert Bartoszynski who worked with Łoś. Robert was a real virtuoso of, and so my main consultant on, probability theory. Because of my interest in linguistics, I also talked a lot with Robert's wife who was a linguist. I remember following some seminars by Sierpinski – he was quite old then, always taken care of by the famous, then young, number

theorist Andrzej Schintzel. Because IMPAN was so well-known worldwide, we had a lot of visitors and this gave me a chance to meet a lot of famous scientists. For example, I met Solomon Marcus when he was visiting Pawlak. I spent a lot of time with him talking about science and many other matters, he also met my parents and my wife. I must have been among the first researchers he introduced to contextual grammars, a topic which I picked up again much later when I worked with Gheorghe Păun (a student of Marcus) on it. Marcus invited me to Bucharest to work together, and I still remember a very nice visit there. Anyhow, I became an admirer of Marcus and remain so still today. We meet from time to time at various events, and I cherish these meetings.

My relationship to Marcus continued also in a different way when during my later years in science, I became a collaborator, mentor, and friend of many Romanian scientists educated/influenced by him. This group includes Lila Kari, Gheorghe Păun, yourself, and Elena, Alexandru Mateescu, and Ion Petre. I was always impressed by the mathematical and human qualities of disciples of Marcus.

My first big new research topic at IMPAN was category theory – I got interested in both pure category theory and its potential to express and investigate computations. Concerning the former, I worked on axioms for the category of relations and this work brought me in contact with Samuel Eilenberg. I was very flattered by his interest in my work. We also remained in contact after I left Poland. He visited me in Utrecht and stayed in our apartment. His main passion outside mathematics was collecting certain types of figurines from Indonesia. Because of the long history of Dutch-Indonesian relationship, The Netherlands was a real gold mine for these figurines. So I visited a lot of "strange places" with him in Utrecht and Amsterdam.

At the beginning of my commuting to Boulder I met Stan Ulam, another famous Polish mathematician. Also, together with Aristid Lindenmayer, we invited Ulam to attend a symposium we organised in The Netherlands (on information processing in biology). Thus I had many conversations with Ulam and was fascinated by him. Mostowski, Eilenberg, and Ulam were typical representatives of the famous old school of Polish mathematics. There was something common (in my perception) to all three of them: they were brilliant, erudite, well-mannered, and had a very good sense of humour (I was certainly telling jokes to all three of them).

I was very much influenced by the paper "Finite automata and their decision problems" by M. Rabin and D. Scott – it was certainly one of the most important papers I read. I started right away working on various problems inspired by it. In particular, I started to develop a theory of multitape automata, this was going very well and I hoped it would become my PhD thesis. Then one day Mostowski brought a manuscript (I think that this was an official report from Harvard, perhaps a PhD thesis) by Arnold Rosenberg on multitape automata, and asked me to look

it up in connection with my own research. I observed that more than half of my results (with many of them already presented at our internal seminar) were covered by Arnold. I even remember making a joke that if Arnold's surname would be also written with "z" (hence "Rozenberg") then ALL my results would be already covered by him! Mostowski explained then to me (he was always very kind and supportive) that in mathematics if you get "good" results and discover later that these results were already proved by good scientists, then you get in this way the best possible confirmation that your research is on a good path. I decided then to switch to research on certain type of regular languages and got my PhD for this work.

This and a number of other events made me realise how isolated we were in Poland (nobody really cared about "us"!), even though through personal connections of Mostowski and others we were in a privileged position. I remember making a resolution then that if I ever got out of Poland, I would "do a lot for the scientific community" as opposed to "doing a lot only for myself". This resolution got strongly implemented when I left Poland. I have devoted a huge amount of my professional time to service for the academic community – this includes my work for EATCS, my work for organising conferences, my work for founding new journals and book series, ... Clearly, my list of publications would be much longer if I would not spend so much time in the service of the scientific community. But, I always remembered my resolution from Poland and really get a lot of satisfaction from serving the community and seeing many positive effects of this service.

To summarise, I was really lucky and privileged to work at IMPAN. It was a real oasis of tranquility: while there, the surrounding reality of the totalitarian political system was nonexistent. The only thing that counted was science, there were no political activities. Clearly, the situation must have been very different for people running the Institute, as they had to deal with the outside world.

CC: Why do you like so much Hieronymus Bosch paintings?

GR: I was always interested in paintings, and during my youth in Poland I was "possessed" by impressionism. I read everything that was accessible to me there about impressionism, looked up all possible albums with reproductions, even had in my room reproductions of van Gogh and Monet hanging on the walls. When I settled in The Netherlands, it was a sheer delight to visit museums here and see real paintings by impressionists as well as to go to Paris to see even more there.

However, one day, just by chance, I bought a book with many reproductions about Bosch, and right away I fell in love with Bosch (and impressionists were moved to the back burner). This love for Bosch only intensified with time.

He is an enigmatic painter in many ways, and therefore a difficult painter for art historians to analyse. Hence, e.g., we know very little about his life, we don't

even know when he was born except for some reasoning which leads to "around 1450" – his funeral took place on August 9, 1516. This on its own is quite an obstacle in analysing his art. Furthermore, no more than 25 of his paintings survived and we are not even sure whether all of them are authentic. He signed only a few of these paintings and none of them is dated.

But what we know for sure is that he was a genius, who went his own way, and was much ahead of his time – much of his creation is of timeless beauty. For me he is a personification of vision and creativity. My admiration for him is very well expressed by Jose de Siguenza (1544–1606) who was a historian, monk, and prior of the monastery of El Escorial (a Spanish royal site close to Madrid). El Escorial was home to many Bosch paintings collected by Phillip II of Spain. Jose de Siguenza wrote that he was amazed that "a single mind could imagine so many things".

The best known of Bosch's paintings is "The Garden of Earthly Delights" in Museo del Prado in Madrid. Many art historians list it as one of the most remarkable paintings ever. For me Prado is the best museum in the world, as they have (in one room!) 5–6 paintings of Bosch (recall that no more than 25 paintings of Bosch exist today). Madrid is my favourite art city as they also have Bosch paintings in Palacio Real and close by in El Escorial. Then on top of it Madrid is famous for its school of card magic!!!

Bosch drawings are less known than his paintings, but his drawings are also extraordinary. Again, no more than 40 of his drawings survived. He almost exclusively used only the pen in his drawings. My favourite drawing by Bosch is "The Wood Has Ears, the Field Eyes", which depicts a larger tree in front of a grove of smaller trees, all set up in a meadow. The drawing shows a number of open eyes embedded in a meadow and two large ears embedded between trees of the grove. There is a later Netherlandish woodcut from 1546 (30 years after Bosch's death), possibly based on Bosch's drawing, which illustrates the same theme, where the inscription says "The field has eyes, the wood has ears, I will see, be silent, and listen." This demonstrates the timeliness of Bosch's art – think about today's concern about privacy in the time of all the electronic media, surveillance cameras, etc. Even more interestingly, at the top of this drawing there is an inscription in Latin which says "For poor is the mind that always uses the ideas of others and invents none of its own". Most probably it was the "official motto" of his workshop, but it surely should be the motto for each researcher!

The larger central tree in this drawing has an owl sitting in a natural hollow opening in it. At this time period in this geographic location (Brabant) owls were a symbol of wickedness and evil spirits. Thus the owl in the center of the drawing was contributing to the intended theme of the drawing.

As a matter of fact, owls appear a lot in paintings of Bosch and also in his drawings. For example, another drawing of Bosch which I like a lot is "Owl's Nest

on a Branch". Also, owls are quite central in "The Garden of Earthly Delights". Bosch was certainly fascinated by owls!

Since I am also fascinated by owls, this makes Bosch art even more dear to me. My interest in owls originated in science, more precisely in my collaboration with Juhani Karhumäki. I was his mentor when we worked on his Ph.D. thesis with Arto Salomaa (by today Juhani is one of the world leaders in combinatorics on words). On one of his working visits to my home in The Netherlands, he brought many pictures of young (baby) owls in their nests. Juhani is also an ornithologist and spends a lot of time during the summers (mostly in June) banding young birds high in their tree nests (sometimes 30–40 meters high!!!). He is also an excellent photographer, so his pictures of young owls were really beautiful.

I fell in love, first with the pictures of owls, and then with owls in general. Started to read a lot about real owls but also about the images and the symbolism of owls in various cultures all over the world. By today I have a collection of over 2000 owls of all sorts: real stuffed owls, ceramic owls, glass owls, metal owls, silver owls, incrustrated owls, ...

A painting by Bosch which is well-known to many magicians is "The Conjurer". It is a beautiful painting depicting a magician performing (most probably at a market).

I need now to make a digression into the history of magic. Unfortunately the history of magic is not so glorious, as magic was often used as an instrument of control and as a skill for cleaning people out of their possessions/money. As examples of the former, one can point out that pharaohs in Egypt had magicians in their entourage who were performing all kinds of tricks which would prove that pharaohs did possess inhuman powers given to them by gods. As examples of the latter, one can point out magicians robbing people out of money at markets by playing "very fair" cups and balls or 3 card monte guessing games. Thus cheating became closely associated with magicians. Magic became a performing art only in the 19th century (with a lot of credit for this transformation given to the famous French magician Robert Houdin). Today magic flourishes as a performing art and is often referred to as the queen of performing arts.

Going back to "The Conjurer" painting, it depicts a magician at one side of a table, with cups there and a small ball kept "professionally" in his right hand, clearly performing the famous cheating game of "cups and balls". A small group of spectators stands at the other side of the table with one of them "central" in this composition. This central spectator bends over the table watching the magician and is totally flabbergasted by the performance, so much so that a green frog jumps out of his gaping mouth (there was a proverb in Brabant at this time saying that you may be so flabbergasted that a frog will jump out of your mouth). While this spectator is so lost in the performance, a thief (perhaps a confederate of the magician) is cleaning him out of money kept in a leather pouch.

This beautiful painting shows Bosch as a keen observer of everyday life, while it also reminds magicians about the not so glorious history of magic.

I would like to add a comment about my love of visual arts. It begun in Poland when I was a teenager, and it was purely "theoretical" in the sense that nobody in my family had any talent for painting. This situation changed dramatically when my son Daniel was born – it was clear already since he was about three years old that drawing and painting were his vocation. Indeed, he became a very well known visual artist – creations of DADARA (his artist name) are amazing. Since neither me nor my wife Maja had any talent for drawing/painting, in my lectures on molecular biology I was giving Daniel as an example of a "beautiful mutation" – his talent came from "nowhere". It turned out that I was wrong: just a few years ago we discovered that Maja has a real talent for painting. In fact Daniel says now that his artistic genes come from her. Thus, when my love for paintings begun in Poland I had just (cheap) *reproductions* of van Gogh and Monet hanging on the walls of my room. Now our home is full of *original* beautiful paintings by DADARA and Maja!

CC: You really love books. Which of them have influenced most your professional life?

GR: I have loved books all my life. My wife said once that I spend my money on books and playing cards! However I remember that when I was a teenager in Poland, there were non-monetary ways to get access to good books. Many good books from before the World War II were not available in bookstores because they were "ideologically wrong". The way to get access to these books was through ...rewriting. One could borrow such an unavailable book (or a hand rewritten copy of it) for a certain period of time, and during this time one would rewrite (a part of) this book. The borrowed book had to be returned, but one would have a handwritten copy that could be read several times. Such a copy could be also exchanged for a handwritten copy of another book. Rewriting a book by hand was very time consuming, so one had to be a real book lover to engage in this way of collecting books.

During my study years and also during my work at IMPAN I profited a lot from the lawless pirating behaviour of the Soviet Union. They were translating scientific books published in the West on a massive scale, without respect for copyrights. Moreover, all Russian books were very cheap in Poland. In this way I read many excellent science books published in the West – without the lawless behaviour of Soviet Union I would not have had access to most of these books!

One of the blessings of working at IMPAN was their mathematics library – certainly the best source of mathematical books and journals in Poland.

Also, the library of the Institute of Foundations of Informatics of Polish Academy of Sciences (Polish acronym: IPIPAN) in Warsaw had a very good li-

brary, especially of computer science and electrical engineering books. This was my library when I was a student. I would sit there whole days, as many of the books there (especially British and American books) could not be moved out of the library. I became a good friend with the young librarian (her name was Lidia Miernicka) and at some point we were doing something illegal (which could have bad consequences for her): when she was closing the library in the evening I was allowed by her to take with me (secretly) a couple of books which had to be back on the shelves when she was opening the library in the morning. This meant that I was studying the books all night and waiting for her to open the library to (secretly) return the books. I was extremely indebted to her. When many, many years later Poland became a noncommunist country and I learned that she was the main librarian of IPIPAN I began to buy books for her library as a way of saying "thank you" for what she did for me when I was a student. When I was lecturing at IPIPAN some time ago, I was shown a wall of shelves filled in with "Rozenberg books" – to see this was very satisfying and emotional for me.

I should also mention that I have a really impressive collection of books on Hieronymus Bosch, perhaps one of the best private collections in The Netherlands. Indeed a lot of money and collecting effort went into establishing this collection, but it is very useful for my studies of Bosch.

As for the books that influenced my professional life, this would be a long list which would require a long time to construct (also because of my bad memory). But on a short call, and somehow ad hoc I would list the following books: "Network Analysis" by Van Valkenburg, "Set Theory" by Kuratowski and Mostowski, "Abelian Categories" by Freyd, "Elements of Mathematical Logic" by Rosenbloom, "Automata Studies" edited by Shannon and McCarthy, "Computability and Unsolvability" by Davis, "Algebraic Structure Theory of Sequential Machines", by Hartmanis and Stearns, "Mathematical Theory of Context-Free Languages" by Ginsburg, "Formal Languages" by Salomaa, "The Language of Life" by Beadle and Beadle, "Dealing with Genes" by Berg and Singer, "Recombination DNA" by Watson, Tooze and Kurtz, "Bioenergetics" by Lehninger, and several books by Peter Atkins on chemistry and thermodynamics.

CC: Many thanks.

Russian Chapter of European Association for Theoretical Computer Science

Edward A. Hirsch*

January 12, 2015

During the 9th International Computer Science Symposium in Russia (CSR-2014) in Moscow, a meeting was held where it was decided to create Russian Chapter of European Association for Theoretical Computer Science, by analogy to, for example, the currently existing Italian and Japanese Chapters of EATCS. The objectives of the new chapter are promoting and encouraging research, publishing, education, meetings (including CSR conferences) related to theoretical computer science, explaining and communicating traditions and current developments of theoretical computer science to other scientists, authorities and the general public. Russian Chapter also intends to cooperate with other associations in the field of theoretical computer science and related fields.

A Statute Committee (Alexander Rubtsov, Nikolai Vereshchagin (chair), Mikhail Vyalyi, Mansur Ziatdinov) then wrote the Chapter Statute, which was approved by EATCS Council. The initial members of the chapter (as of December 24, 2014) are those members of EATCS who stated their Russian affiliation in their EATCS membership information and expressed their willingness to accept the Statute and join the chapter. In the future, any member of EATCS (with any affiliation) will be able to join the chapter upon either publishing at least one paper in a series sponsored by EATCS (or its Russian Chapter), or presenting a brief recommendation letter from two members of the chapter or one member of its Council (of course, if s/he supports the objectives of the chapter).

The next step will be elections of the President, the Vice-President and the Public Relations Officer organized by the Statute Committee.

We are looking forward to new people (both current and new members of the EATCS) joining the chapter! To learn more about the chapter and its contact information, please visit its web site `http://logic.pdmi.ras.ru/~rceatcs/` .

*Steklov Institute of Mathematics at St.Petersburg, Russia

Report on CPM 2014

25th Annual Symposium on Combinatorial Pattern Matching
Moscow, Russia, June 16-18, 2014

Mikhail Roytberg

I took part in CPM Symposiums only once, thus the choice of the city for CPM 2014 gave me a nice opportunity to listen to a lot of interesting talks and to see many old friends without stopping my habitual life. The number 25 in the title of the Symposium suggested the organizers to make the list of invited speakers very close to those of the first CPM that was held in 1990. I think it was a good idea! By the way, preparing this text I discovered for myself the CPM archive http://www.cs.ucr.edu/~stelo/cpm; it is a fruitful source of information for those who are interested in the history of computer science. Despite its name the Symposium represents the state of the art not only in pattern matching but in various areas of theoretical and applied computer science, e.g. data compression, stringology and bioinformatics.

During 3 days in the conference hall of Yandex company were presented 28 contributed talks that were selected from 51 submissions by the Program Committee (Chairs: Alexander Kulikov, Pavel Pevzner); the speakers represented 15 countries. The talks are available at http://cpm2014.hse.ru/videos, thanks to the Organizing Committee (Chair: Sergei Kuznetsov). The proceedings are available in the LNCS series of Springer, Volume 8486:
http://www.springer.com/computer/image+processing/book/978-3-319-07565-5

The morning session of the first day paid tribute to 25th CPM Anniversary and included three invited talks (Alberto Apostolico "Sequence Comparison in the Time of Deluge", Maxime Crochemore "Repeats in Strings", Gene Myers "What's behind BLAST") that reviewed three important issues of sequence analysis. The forth invited talk (Udi Manber "How to Think Big") was given on the next morning, the speaker communicated with his audience using most advanced communication technologies.

The contributed talks were presented during other four sessions of the symposium, each session consisting of two parts. At the evening session of the 1st day were given talks on repeats and palindromes (1st part) and data compression (2nd part). The morning session of the 2nd day was devoted to indexing and pattern matching in its classical formulation. After lunch the participants had an opportunity to visit the most famous Moscow sites including the Kremlin. At

the morning session of the last day the pattern matching line was continued (1st part) and graph-related issues were considered (2nd part). The final session was devoted to vocabularies, grammars, substrings and prefixes.

One can see that the talks cover all sequence-related areas of computer science (may be except sequence alignment) and some important issues beyond sequence analysis. I (and I think all participants) enjoyed the conference and I would like to give my deepest thanks to organizers and speakers, and wish long live to CPM Symposiums. See you at CPM 2015!

Report on UCNC 2012
the 11th International Conference on Unconventional Computation and Natural Computation

Susan Stepney

Prior to 2012, this conference series was known as Unconventional Computation (UC); this year the name changed to UCNC, reflecting the close link between the two disciplines.

The 11th International Conference on Unconventional Computation and Natural Computation (UC 2012) took place at the University of Orléans, France, 3–7 September 2012. It was organised by the Laboratoire d'Informatique Fondamentale d'Orléans and was held at the computer science building on the campus of *La Source* of the Université d'Orléans. The conference received support from: LIFO, the Laboratoire d'Informatique Fondamentale d'Orléans; the University of Orléans; the City of Orléans; Conseil Général du Loiret; Région Centre; GdR Informatique et Mathématiques; CNRS; Inria.

The fully international complement of authors came from all parts of the globe: Austria, Canada, Colombia, France, Germany, Hungary, Iran, Ireland, Italy, Japan, Moldova, Netherlands. New Zealand, Norway, Poland, Romania, South Korea, Spain, Sweden, UK, and USA.

The invited plenary session speakers (in alphabetical order) and their talk titles were: Paola Bonizzoni (University of Milano-Bicocca, Italy) "The Holy Grail: Finding the Genetic Bases of Phenotypic Characters"; Cristian S. Calude (University of Auckland, New Zealand) "Inductive Complexity of P versus NP Problem"; René Doursat (GEB, Universidad de Málaga, Spain) "Advances in Embryomorphic Engineering"; Jack Lutz (Iowa State University, USA) "Finite State Dimensions" (in a change to the advertised title); Maurice Margenstern (Université de Lorraine, France) "Universality and the Halting Problem for Cellular Automata in Hyperbolic Spaces: The Side of the Halting Problem".

There were two extended tutorials, each comprising three 1 hour sessions. "MGS

and Spatial Computing", by Jean-Louis Giavitto (IRCAM, Paris, France) and Olivier Michel, Antoine Spicher (LACL, Université de Paris Est – Créteil, France) was an in-depth introduction to their topological programming paradigm, including the underlying mathematical concepts, the language, and applications from morphogenesis to music. "An Introduction to Tile-Based Self-assembly" by Matthew Patitz (University of Arkansas, USA) was a detailed look at a wide range of results from two dimensional tiling theory and self-assembly of DNA tiles.

The full conference comprised these plenaries and tutorials, together with the scientific programme of technical presentations of the published papers, and a poster session where the authors had the opportunity to make a short informal presentation of their work.

Proceedings of UCNC 2012 are published in the Springer series as LNCS volume 7445 (ISBN 978-3-642-32893-0). The volume contains abstracts and extended abstracts of the invited papers and tutorials, 14 refereed 12 page full papers, and 6 poster paper abstracts.

In association with the main technical conference, there were four parallel associated workshops on related unconventional topics: 5th International workshop on Complex Systems Modelling and Simulation (CoSMoS 2012); 2nd COBRA workshop on Biological and Chemical Information Technologies (BioChemIT 2012); 1st International Workshop on Information Physics and Computing in Nano-scale Photonics and Materials (IPCN 2012); 2nd Workshop on Foundations of Quantum Information (FounQI 2).

In addition, Gilles Dowek (INRIA, Paris, France) gave a public lecture as part of the Turing Centenary celebrations. He spoke on "A Two-Dimensional Programming Language for Two-Dimensional Data". Over the course of a fascinating hour, he gradually built up a Universal Turing Machine in a simple two dimensional graphical language, in a way that made it crystal clear how UTMs work.

Because of the parallel conference and workshop sessions, I attended only a selection of the presentations. Particular highlights for me, in addition to Dowek's talk, include the following, both on morphogenesis, one of my interests.

René Doursat's plenary on Embryomorphic Engineering covered a wide range of issues in complex systems, emergence, self-organisation, and morphogenesis, building up to the denouement: engineering the evolution of computational "seeds", and the growth from these seeds, of complex ALife systems.

Jean-Louis Giavitto and Antoine Spicher's MGS tutorial demonstrated an extremely elegant approach to unifying many of the different kinds of rewriting

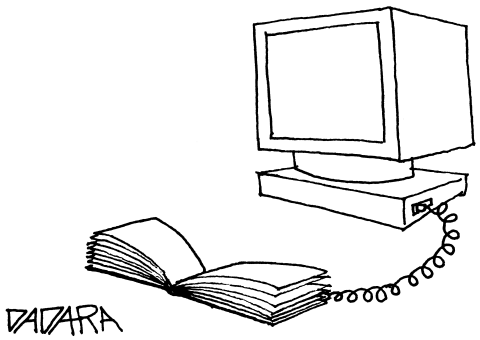
systems that form a sub-branch of unconventional computation. It provides a topological approach to a range of dynamical systems, getting a grip on "dynamical systems with dynamical structure", essential for modelling, simulating and engineering morphogenesis.

The social programme allowed a choice: canoeing on the Loire, or a visit to the Château de Chambord. I chose the latter. A visit to Chambord is a marvellous experience, particularly in the fine weather that we enjoyed. We were introduced to the magnificent architecture and its accompanying history by an enthusiastic and articulate guide; afterwards, I researched a couple of the stories she had told, and discovered that they might have been slightly exaggerated! We returned to Orléans for a walking tour of the city, which has many impressive buildings and many excellent statues of Jeanne d'Arc. After the afternoon excursions, the two groups the met up for the conference dinner.

Many thanks for an excellent event go to the local organisers: Florent Becker (local chair), Jérôme Durand-Lose, Bastien Le Gloannec, Mathieu Liedloff, Nicolas Ollinger, Anthony Perez, and Maxime Senot.

Next year's UCNC takes place 1–5 July 2013, in Milan, Italy.

Announcements



IWOCA 2015 in Verona, Italy

Zsuzsanna Lipták Bill Smyth

The **26th International Workshop on Combinatorial Algorithms (IWOCA 2015)** will take place in Verona, Italy, from 5-7 October 2015. Verona is one of the most beautiful medieval cities in a country beautiful medieval cities; not for nothing is its historic centre a UNESCO world heritage site. The conference will take place right in the old town, in a 17th c. building, 5-10 minutes walk from the river Adige, from the Arena (Verona's fully conserved Roman amphitheater), from the subterranean city remains, from the Roman theatre, from the beautiful medieval cathedral, to name just a few of the stunning touristic sites.

IWOCA arises out of a quarter-century of history: International Workshop since 2007, Australasian Workshop 1989–2006. Previous IWOCA resp. AWOCA meetings have been held in Australia, Canada, Czech Republic, France, Indonesia, India, Japan, South Korea, UK, and the USA. IWOCA 2015 continues the long and well-established tradition of encouraging high-quality research in theoretical computer science and providing an opportunity to bring together specialists and young researchers working in the area. The scientific program will include invited lectures, accepted contributing talks, posters, and a problem session.

We hope for many high-quality contributions on the topics of the conference, which include (but are not restricted to): algorithms and data structures (including sequential, parallel, distributed, approximation, probabilistic, randomized, and on-line algorithms), algorithms on strings and graphs, applications (bioinformatics, music analysis, networking, and others), combinatorics on words, combinatorial enumeration, combinatorial optimization, complexity theory, computational biology, compression and information retrieval, cryptography and information security, decompositions and combinatorial designs, discrete and computational geometry, graph drawing and labelling, graph theory. For more information, see the website of the conference: `http://iwoca2015.di.univr.it`.

E. W. Beth Dissertation Prize 2015

Call for Nominations

Deadline: April 27th, 2015.

Since 2002, FoLLI (the Association for Logic, Language, and Information, `http://www.folli.info`) has awarded the E.W. Beth Dissertation Prize to outstanding dissertations in the fields of Logic, Language, and Information. We invite submissions for the best dissertation which resulted in a Ph.D. degree awarded in 2014. The dissertations will be judged on technical depth and strength, originality, and impact made in at least two of three fields of Logic, Language, and Computation. Interdisciplinarity is an important feature of the theses competing for the E.W. Beth Dissertation Prize.

Who qualifies.

Nominations of candidates are admitted who were awarded a Ph.D. degree in the areas of Logic, Language, or Information between January 1st, 2014 and December 31st, 2014. Theses must be written in English; however, the Committee accepts submissions of English translations of theses originally written in other languages, and for which a PhD was awarded in the preceding two years (i.e. between January 1st, 2012 and December 31st, 2013). There is no restriction on the nationality of the candidate or on the university where the Ph.D. was granted.

Prize.

The prize consists of:

- a certificate
- a donation of 2500 euros provided by the E.W. Beth Foundation
- an invitation to submit the thesis (or a revised version of it) to the FoLLI Publications on Logic, Language and Information (Springer). For further information on this series see the FoLLI site.

How to submit.

Only electronic submissions are accepted. The following documents are required:

1. The thesis in pdf format (ps/doc/rtf not accepted).

2. A ten-page abstract of the dissertation in pdf format.

3. A letter of nomination from the thesis supervisor. Self-nominations are not admitted: each nomination must be sponsored by the thesis supervisor. The letter of nomination should concisely describe the scope and significance of the dissertation and state when the degree was officially awarded.

4. Two additional letters of support, including at least one letter from a referee not affiliated with the academic institution that awarded the Ph.D. degree.

All documents must be submitted electronically (preferably as a zip file) to Ian Pratt-Hartmann (`ipratt@cs.man.ac.uk`). Hard copy submissions are not allowed. In case of any problems with the email submission or a lack of notification within three working days, nominators should write to Ian Pratt-Hartmann. The prize will be awarded at the ESSLLI summer school in Barcelona.

Important dates:
Deadline for Submissions: April 27th, 2015.
Notification of Decision: July 6th, 2015.
ESSLLI summer school: August 3rd – 14th, 2015.

Committee :

- Raffaella Bernardi (Trento)
- Johan Bos (Groningen)
- Julian Bradfield (Edinburgh)
- Wojciech Buszkowski (Poznan)
- Michael Kaminski (Technion, Haifa)
- Marco Kuhlmann (Linkoping)
- Larry Moss (Bloomington)
- Valeria de Paiva (Nuance Communications)
- Ian Pratt-Hartmann (chair) (Manchester)
- Ruy de Queiroz (Recife)

- Mehrnoosh Sadrzadeh (Queen Mary, London)
- Rineke Verbrugge (Groningen)

European Association for Theoretical Computer Science

E A T C S

EATCS

HISTORY AND ORGANIZATION

EATCS is an international organization founded in 1972. Its aim is to facilitate the exchange of ideas and results among theoretical computer scientists as well as to stimulate cooperation between the theoretical and the practical community in computer science.

Its activities are coordinated by the Council of EATCS, which elects a President, Vice Presidents, and a Treasurer. Policy guidelines are determined by the Council and the General Assembly of EATCS. This assembly is scheduled to take place during the annual **I**nternational **C**olloquium on **A**utomata, **L**anguages and **P**rogramming (ICALP), the conference of EATCS.

MAJOR ACTIVITIES OF EATCS

- Organization of ICALP;
- Publication of the "Bulletin of the EATCS;"
- Award of research and academic career prizes, including the EATCS Award, the Gödel Prize (with SIGACT), the Presburger Award, the Nerode Award (joint with IPEC) and best papers awards at several top conferences;
- Active involvement in publications generally within theoretical computer science.

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: CIAC (Conference of Algorithms and Complexity), CiE (Conference of Computer Science Models of Computation in Context), DISC (International Symposium on Distributed Computing), DLT (International Conference on Developments in Language Theory), ESA (European Symposium on Algorithms), ETAPS (The European Joint Conferences on Theory and Practice of Software), LICS (Logic in Computer Science), MFCS (Mathematical Foundations of Computer Science), WADS (Algorithms and Data Structures Symposium), WoLLIC (Workshop on Logic, Language, Information and Computation), WORDS (International Conference on Words).

Benefits offered by EATCS include:

- Subscription to the "Bulletin of the EATCS;"
- Access to the Springer Reading Room;
- Reduced registration fees at various conferences;
- Reciprocity agreements with other organizations;
- 25% discount when purchasing ICALP proceedings;
- 25% discount in purchasing books from "EATCS Monographs" and "EATCS Texts;"
- Discount (about 70%) per individual annual subscription to "Theoretical Computer Science;"
- Discount (about 70%) per individual annual subscription to "Fundamenta Informaticae."

(1) THE ICALP CONFERENCE

ICALP is an international conference covering all aspects of theoretical computer science and now customarily taking place during the second or third week of July. Typical topics discussed during recent ICALP conferences are: computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of logic programming, theorem proving, software specification, computational geometry, data types and data structures, theory of data bases and knowledge based systems, data security, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

SITES OF ICALP MEETINGS:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, UK 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Nafplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, UK 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologne, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006
- Wrocław, Poland 2007
- Reykjavik, Iceland 2008
- Rhodes, Greece 2009
- Bordeaux, France 2010
- Zürich, Switzerland 2011
- Warwick, UK 2012
- Riga, Latvia 2013
- Copenhagen, Denmark 2014

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for *rapid* publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D. theses;
- Entertainments and pictures related to computer science.

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at `http://www.eatcs.org/bulletin`. Questions and proposals can be addressed to the Editor by email at `bulletin@eatcs.org`.

(3) OTHER PUBLICATIONS

EATCS has played a major role in establishing what today are some of the most prestigious publication within theoretical computer science.

These include the *EATCS Texts* and the *EATCS Monographs* published by Springer-Verlag and launched during ICALP in 1984. The Springer series include *monographs* covering all areas of theoretical computer science, and aimed at the research community and graduate students, as well as *texts* intended mostly for the graduate level, where an undergraduate background in computer science is typically assumed.

Updated information about the series can be obtained from the publisher.

The editors of the EATCS Monographs and Texts are now M. Henzinger (Wien), J. Hromkovic (Zürich), M. Nielsen (Aarhus), G. Rozenberg (Leiden), A. Salomaa (Turku). Potential authors should contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:
Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands
who acknowledges EATCS membership and forwards the order to Springer-Verlag.

The journal *Theoretical Computer Science*, founded in 1975 on the initiative of EATCS, is published by Elsevier Science Publishers. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.

The Editor-in-Chief of the journal currently are G. Ausiello (Rome) and D. Sannella (Edinburgh).

ADDITIONAL EATCS INFORMATION

For further information please visit `http://www.eatcs.org`, or contact the President of EATCS:
Prof. Dr. Luca Aceto,
School of Computer Science
Reykjavik University
Menntavegur 1 IS-101 Reykjavik, Iceland
Email: `president@eatcs.org`

EATCS MEMBERSHIP

DUES

The dues are € 30 for a period of one year (two years for students). A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.

In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for € 25 per year. We also offer a five-euro discount on the EATCS membership fee to those who register both to the EATCS and to one of its chapters. Additional € 25 fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe.

HOW TO JOIN EATCS

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website `www.eatcs.org`, where you will find an online registration form and the possibility of secure online payment. Alternatively, a subscription form can be downloaded from `www.eatcs.org` to

be filled and sent together with the annual dues (or a multiple thereof, if membership for multiple years is required) to the **Treasurer** of EATCS:

Prof. Dr. Dirk Janssens,
University of Antwerp, Dept. of Math. and Computer Science
Middelheimlaan 1, B-2020 Antwerpen, Belgium
Email: `treasurer@eatcs.org`, *Tel: +32 3 2653904, Fax: +32 3 2653777*

The dues can be paid (in order of preference) by VISA or EUROCARD/MASTERCARD credit card, by cheques, or convertible currency cash. Transfers of larger amounts may be made via the following bank account. Please, add € 5 per transfer to cover bank charges, and send the necessary information (reason for the payment, name and address) to the treasurer.

Fortis Bank, Jules Moretuslei 229, B-2610 Wilrijk, Belgium
Account number: 220–0596350–30–01130
IBAN code: BE 15 2200 5963 5030, SWIFT code: GEBABE BB 18A